The thesis committee for Vamsi Krishna Nadimpalli certifies that this is the approved version of the following thesis

An average cost Markov decision process model to decide when to challenge a call in a tennis match

APPROVED BY

SUPERVISING COMMITTEE:

John J. Hasenbein, Supervisor

J. Eric Bickel

An average cost Markov decision process model to decide when to challenge a call in a tennis match

by

Vamsi Krishna Nadimpalli, B.Tech.

THESIS

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN ENGINEERING

THE UNIVERSITY OF TEXAS AT AUSTIN

August 2010

Dedicated to my mother Krishnaveni.

Acknowledgments

I am very grateful to Prof. John Hasenbein for giving me the opportunity to do research under his supervision and providing me with expert suggestions and guidance throughout the course of this work. The courses taught by him created a lot of interest in stochastic processes and provided me the motivation to do research in this area. I would like to thank Dr. Bickel for accepting to be the reader for this work and Dr. Morton for helping me out when ever I needed help with coding my mathematical model. Also, I would like to thank Ms. Diana Ziegler and Mr. David Justh for providing me excellent administrative support and High Performance Lab personnel for providing me the required computational facilities.

Finally, I would like to thank my family and friends for their love, encouragement and emotional support for this endeavor and for making my stay at University of Texas at Austin a memorable one.

An average cost Markov decision process model to decide when to challenge a call in a tennis match

Vamsi Krishna Nadimpalli, MSE
The University of Texas at Austin, 2010

Supervisor: John J. Hasenbein

In a standard tennis match each player has an unlimited opportunity to challenge an umpire's call, but if three incorrect challenges are made in a set he is not allowed to challenge anymore in that set. If the set goes into a tie break the limit on incorrect challenges increases by one. These limited incorrect challenges are not carried over from one set to another. So this is kind of a limited resource available to the player and if he knows how to use this resource in a best possible way, there is a scope for increasing his overall chances of winning a match. With the motive of gaining insight on when to challenge a call, we have modeled a single game in a tennis match as a Markov decision process. We have also studied the impact of variables like player's probability of winning a point, the player's perception of the challengability of a call and proportion of challengable calls on the decision making process.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Mathematical modeling as applied to decision making in sports has always been an area that has attracted a lot of attention from researchers. Numerous mathematical models have been developed as applicable to sports in general and tennis in particular. To mention some, Newton and Keller (2005) derived formulas for the probabilities of winning a game, set and match of tennis by modeling it as a Markov chain and by using hierarchal recurrence relations. This work assumes player's probability of winning a point is constant througout a match. Newton and Aslam (2009) developed a Markov chain model to obtain the probability density function (pdf) of winning a match in tennis. This work considered that the probability of winning a point varies from point to point and match to match. Morris (1977) studied the importance of different points in tennis.

To the best of our knowledge, no mathematical models have been developed to determine when to challenge a call in tennis. In this work we address this problem by modeling a game of tennis as a Markov decision process. Similar modelling has been done earlier by researchers as applied to different sports. Clarke (1988) developed a dynamic programming formulation, which

allows one to calculate the optimal scoring rate at any stage in an inning of a game of cricket. Norman (1985) formulated a dynamic program to decide whether to serve fast or slow at each stage in a game of tennis. Kohler (1982) came up with an absorbing state stochastic dynamic program, to decide where to aim in each play in a game of darts.

With this background, the next two sections are dedicated to giving a general introduction to Markov decision processes and to the rules of tennis, as they are essential for understanding our model.

1.1 Markov decision processes

Markov decision processes (MDPs) provide a mathematical framework for modeling situations where decisions are made in stages. The outcome of these decisions may not be completely predictable but can be anticipated to some extent before the next decision is made. The objective is to minimize a certain cost or maximize a certain reward. In such situations each decision cannot be viewed in isolation since one must balance the desire for low present cost with undesirability of high future costs. The MDP modeling framework captures this tradeoff. At each stage, these models rank decisions based on the sum of the present cost and expected future cost, assuming optimal decision making for subsequent stages.

MDPs were developed at least as early as the 1950s (cf. Bellman 1957). Presently, they are used in various areas including robotics, automated control, economics and manufacturing.

1.1.1 Basic MDP model

The basic model has two principal features: (1) an underlying discrete-time dynamic system, and (2) a cost function that is additive over time. The dynamic system expresses the evolution of some variables, the system's *state* under the influence of decisions made/action taken at discrete instances of time. The system has the form

$$x_{k+1} = f_k(x_k, u_k, w_k), \qquad k = 0, 1, ..., N-1,$$

where

k	indexes discrete time,
x_k	is the state of the system at time k,
u_k	is the action or decision variable to be selected at time k,
w_k	is a random variable representing a noise or disturbance,
N	is the horizon or number of times action is applied,

and f_k is the function that describes the system and in particular the mechanism by which the state is updated.

The cost function is additive in the sense that the cost incurred at time k, denoted by $g_k(x_k, u_k, w_k)$, accumulates over time. The total cost is

$$g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k, w_k),$$

where $g_N(x_N)$ is a terminal cost incurred at the end of the process. The cost is generally a random variable because of the presence of w_k. We therefore formulate the problem as an optimization of expected cost

$$E\left\{ g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k, w_k) \right\},$$

where the expectation is with respect to the joint distribution of the random disturbances. The optimization is over the actions $u_0, u_1, \ldots, u_{N-1}$, but some qualification is needed here; each action is selected with knowledge of the current state x_k, but without knowledge of future states. The evolution of states over time can be specified by transition probabilities.

For our purposes, it is sufficient to assume that the state space of the system is finite. We represent the state x_k with a finite-dimensional real vector. The set of actions u_k available in each state x_k is also finite. We define the system dynamic via $p_{ij}(u, k)$, which is the probability at time k that the next state will be j, given that the current state is i, and the action taken is u, i.e.,

$$p_{ij}(u, k) = P\{x_{k+1} = j | x_k = i, u_k = u\}.$$

This type of state transition can also be described in terms of discrete-time system equation

$$x_{k+1} = w_k,$$

where the probability distribution of the random parameter w_k is

$$P\{w_k = j | x_k = i, u_k = u\} = p_{ij}(u, k), \quad i, j \in S, \quad u \in C.$$

Here S is the state space and C is the action space.

1.1.2 Finite horizon MDP

If the time horizon N described in the basic model earlier is finite then that system is a finite horizon MDP. We are given a discrete-time dynamic

system

$$x_{k+1} = f_k(x_k, u_k, w_k), \qquad k = 0, 1, ..., N-1,$$

where the state x_k is an element of the space S_k, the action u_k is an element of a space C_k, and the random disturbance w_k is an element of the space D_k.

The action u_k is constrained to take values in a given non-empty subset $U(x_k) \subset C_k$, which depends on the current state x_k; that is, $u_k \in U_k(x_k)$ for all $x_k \in S_k$ and k.

The random disturbance w_k is characterized by a probability distribution $P_k(\cdot|x_k, u_k)$ that may depend explicitly on x_k and u_k but not on values of prior disturbances $w_{k-1}, ..., w_0$.

We consider the class of policies that consist of sequence of functions

$$\pi = \{\mu_0, ..., \mu_{N-1}\},$$

where μ_k maps states x_k into actions $u_k = \mu_k(x_k)$ and is such that $\mu_k(x_k) \in U_k(x_k)$ for all $x_k \in S_k$. Such policies are called *admissible policies*.

Given an initial state x_0 and an admissible policy $\pi = \{\mu_0, ..., \mu_{N-1}\}$, the states x_k and disturbances w_k are random variables with distributions defined through the system equation

$$x_{k+1} = f_k(x_k, \mu_{x_k}, w_k), \qquad k = 0, 1, ..., N-1.$$

Thus, for given functions $g_k, k = 0, 1, ..., N$, the expected cost of π

starting at x_0 is

$$J_\pi(x_0) = E\left\{ g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, \mu_k(x_k), w_k) \right\},$$

where the expectation is taken over the random variables w_k and x_k. An optimal policy π^* is the one that minimises this cost; that is

$$J_{\pi^*}(x_0) = \inf_{\pi \in \Pi} J_\pi(x_0),$$

where Π is the set of all admissible policies. In our model an optimal policy exists because we assume a finite state space and finite control space.

In general, the optimal policy π^* may depend on the initial state x_0. However, it is possible most of the time to come up with a policy that is simultaneously optimal for all initial states. The optimal cost depends on x_0 and is given by the expression below:

$$J^*(x_0) = \inf_{\pi \in \Pi} J_\pi(x_0).$$

We can view J^* as a function that assigns an optimal cost $J^*(x_0)$ to each initial state x_0. This function is typically referred to as the *optimal value function* or *optimal cost function*.

1.1.3 Infinite horizon MDP

If the time horizon N described in the basic model earlier is infinite then that system is an infinite horizon MDP. The remaining set up is exactly the same except for the time horizon. Infinite horizon MDP problems can be classified broadly into two types, depending on the objective as listed below:

1. Discounted problems

2. Average cost per stage problems.

The following section gives a basic idea about average cost problems as our model uses this framework.

1.1.3.1 Average cost per stage problems

Given an initial state x_0 we want to find a policy $\pi = \{\mu_0, \mu_1, \ldots\}$ which minimizes the cost function defined as

$$J_\pi(x_0) = \lim_{N\to\infty} \frac{1}{N} E\left\{ \sum_{k=0}^{N-1} g_k(x_k, \mu_k(x_k), w_k) \right\}.$$

We can come up with an optimal cost J^* which maps the intial state x to an optimal cost $J^*(x)$. The expression can be shown as below:

$$J^*(x) = \inf_{\pi\in\Pi} J_\pi(x).$$

A background of discrete time Markov chains (DTMCs) is very essential to understand average cost per stage problems. A quick summary about DTMCs is given with the help of the Figure 1.1 below. The nodes in the figure represent the states and the arcs represent the possible transitions from one state to another. In figure 1.1, we can see that starting from state 1, we can reach state 2 and starting from 2 we can reach 1. So states 1 and 2 are said to *communicate*. State 3 leads to 1, but as 1 does not lead to 3, 1 and 3 do not communicate. Starting from 1 we will return with probability 1 and hence it

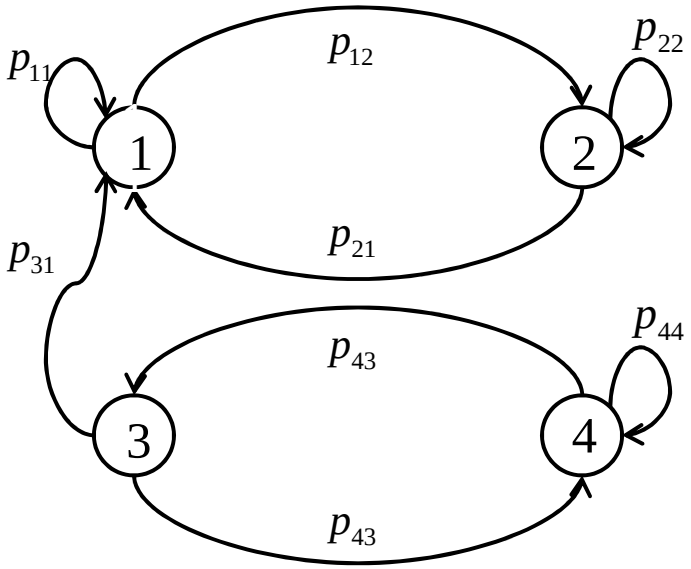

Figure 1.1: Finite state Markov chain

is called a *recurrent* state. Starting from 3 we return with probability strictly less than 1 and hence it is called a *transient* state. As 1 and 2 (respectively 3 and 4) communicate they are of the same type. Therefore, 2 is recurrent and 4 transient. The states which communicate with each other form a subset called *class*. A class is said to be recurrent (respectively transient) if its elements are recurrent (respectively transient). A recurrent class R is an absorbing class in the sense that no element outside R can be reached by any of the elements of R.

With this background about Markov chains we now describe the optimality equations of the average cost Markov decision process models. We have

$$J^*(i) = \min_{u \in U(i)} \left\{ \sum_{j=1}^{n} p_{ij}(u) J^*(j) \right\}, \qquad i = 1, 2, ..., n, \tag{1.1}$$

and

$$J^*(i) + h^*(i) = \min_{u \in \bar{U}(i)} \left\{ g(i,u) + \sum_{j=1}^{n} p_{ij}(u) h^*(j) \right\}, \qquad i = 1, 2, ..., n, \tag{1.2}$$

where $\bar{U}(i)$ is the set of actions attaining the minimum in (1.1) and n is the total number of states. Recall that $J(i)$ is the expected average cost in an intial state i. We can interpret $h(i)$ as the expected average cost in an initial state i relative to some base state j. If J^* and h^* satisfy the pair of equations (1.1) and (1.2), then J^* is the optimal average cost vector. If $\mu^*(i)$ attains the minimum in equations (1.1) and (1.2) for each i, then the stationary policy μ^* is optimal.

Under certain conditions the optimal average cost will be same for all the initial states. One sufficient condition for this to be true is known as the "weak accessibility (WA)" condition. We say that the WA condition holds if the set of states S (state space) can be partitioned into sets S_1 and S_2 such that:

1. Under every stationary policy all the states in S_1 are transient.

2. For any $i, j \in S_2$, i is accessible from j.

A notion related to WA is that of a unichain policy. A stationary policy is called *unichain* if the underlying Markov chain has a single recurrent class

and possibly some transient states. If all stationary policies are unchain then the WA condition holds and if the WA condition holds, there exists an optimal stationary policy that is unichain. If the WA condition does not hold, then the MDP is called *multichain*. In this case the optimal value generally depends on the initial state, and there are multiple recurrent classes under the optimal policy. Our model fits into this category, and thus the multichain case will be our focus.

In order to solve an MDP different approaches can be used depending on the kind of the problem and they are listed below:

1. Value iteration
2. Policy iteration
3. Linear programming.

The following section describes the linear programming approach to solving a multi-chain average cost Markov decision process.

1.1.4 Linear Programming approach

Assuming the reader is familiar with the linear programming basics including duality theory, the following linear program produces the optimal values for the corresponding MDP. The optimal policy can then be derived

directly from these values. The linear program is

$$\max \sum_{i=1}^{n} \beta_i J(i)$$

subject to

$$\begin{aligned} J(i) &\leq \sum_{j=1}^{n} p_{ij}(u) J(j), \quad i = 1, \ldots, n, \quad u \in U(i), \\ J(i) + h(i) &\leq g(i,u) + \sum_{j=1}^{n} p_{ij}(u) h(j), \quad i = 1, \ldots, n, \quad u \in U(i), \end{aligned}$$

where the β_i are some positive scalars with $\sum_{i=1}^{i=n} \beta_i = 1$. The corresponding dual program is

$$\min \sum_{i=1}^{n} \sum_{u \in U(i)} q(i,u) g(i,u)$$

subject to

$$\begin{aligned} \sum_{u \in U(j)} q(j,u) &= \sum_{i=1}^{n} \sum_{u \in U(i)} q(i,u) p_{ij}(u), \quad j = 1, \ldots, n, \\ \sum_{u \in U(j)} q(j,u) + \sum_{u \in U(j)} r(j,u) &= \beta_j + \sum_{i=1}^{n} \sum_{u \in U(i)} r(i,u) p_{ij}(u), \quad j = 1, \ldots, n, \\ q(i,u), r(i,u) &\geq 0, \quad i = 1, \ldots, n, \quad u \in U(i). \end{aligned}$$

The optimal policy can actually be derived in a more straightforward manner from the solution to the dual program as we now describe. Consider

an optimal solution $\{q^*(i,u), r^*(i,u) \mid i = 1,2,...n,\ u \in U(i)\}$ to the above dual program and the sets of states

$$\begin{aligned} I^* &= \left\{ i| \sum_{u \in U(i)} q^*(i,u) > 0 \right\}, \\ \bar{I}^* &= \{i|\, i \notin I^*\}. \end{aligned}$$

The following (possibly randomized) policy

$$\begin{aligned} P(u|i) &= \frac{q^*(i,u)}{\sum_{u \in U(i)} q^*(i,u)} \qquad \text{if} \qquad i \in I^*, \\ &= \frac{r^*(i,u)}{\sum_{u \in U(i)} r^*(i,u)} \qquad \text{if} \qquad i \in \bar{I}^*, \end{aligned}$$

where $P(u|i)$ is the probability that a player takes action u when he is in state i, is optimal in general.

In the above linear program states belonging to the set I^* are recurrent and the states belonging to the set $\bar{I}^*$ are transient.

1.2 Tennis

1.2.1 Scoring in tennis

Tennis is either a two player (singles) or a four player (doubles) sport. For modeling purposes in this work we considered only a two player match and hence from hereon we discuss all the rules as applied to a two-player competition. Scoring in a tennis match has a nested structure where in order to win a match a player has to win sets and in order to win sets he has to win games and to win games he has to win points. Thus, an intuitive approach to

mathematically model a tennis match is to start with points and then extend it to games and then to sets.

The most common format at the professional level is either a three-set contest where the first player to win two sets is declared the winner or a five-set contest where the first player to win three sets is the winner. The player who puts the ball into play for the first point is called the *server* and other player is called the *receiver*.

There are two different systems for scoring in a set. They are the *advantage set* and the *tie-break set*. In an advantage set, a player who wins six games with a margin of two games over his opponent wins that set. This might lead to very lengthy matches because the set will continue until this margin is achieved. In a tie-break set a player who wins six games with a margin of two games over his opponent wins that set and if both the players win six games each a tie-break game will be played. The advantage system has been in limited use after the advent of tie-break system. These days only the final set in singles matches of the Australian Open, the French Open, and Wimbledon and in matches in Davis Cup ties are contested under the advantage system.

The server alternates between games in a set: one player serves the first game, the other player the second game, and so on. In a regular game, one player serves for all the points. However, in a tiebreaker game one player serves the first point, the other player serves the next two points, the serve changes again for the next two points, and so on until a player has won at least seven points and has a lead of at least two points.

Scoring in a *standard game* is represented below with the server's score being called first:

- Zero points → Love.
- First point → 15.
- Second point → 30.
- Third point → 40.
- Fourth point → Game.

For example if the server is at three points and the receiver is at two points the score would be 40-30. When both the players win three points the game enters into *deuce*. After deuce, the score is *advantage in* if the server wins the next point. If the server also wins the next point from advantage in then he wins the game. If the receiver wins next point from deuce then the score would be *advantage out* and if he also wins the next point then the receiver wins the game. If the server wins a point from advantage out or if the receiver wins a point from advantage in the game again enters deuce and will continue as such until one of the players has a two point lead. If the serving player is one point away from winning a game, the point is referred to as a *game-point*; if the nonserving player is one point away from winning a game, the point is referred to as a *break-point*.

Scoring in a tie-break game is different from a standard game. In a tie-break game, points are scored 0, 1, 2, 3, etc. The player who wins a minumum

of seven points with a lead of two points over his opponent wins the game and the set. The game will continue until this lead is achieved.

1.2.2 Point challenge system in tennis

In most of the major tournaments today, the player has an unlimited opportunity to challenge as long as he is not incorrect three times in a set. If he incorrectly challenges three times in a set he has to wait until the next set to challenge again. In a set goes into a tie-break, and additional challenge is allowed. These limited incorrect challenges may not be carried over from one set to another. This type of point challenge system is also referred to as the *3 plus 1* system.

During a tennis match, if a shot returned by a player lands outside the boundary line of the court the line umpire calls it to be *out*. If the ball lands inside the boundary line, the umpire does not make a call, but it is understood to be called *in*. When the umpire calls out, the player responsible for it loses a point. When the umpire calls in and the receiver failed to return the ball then he loses a point. If the player is not satisfied with the line umpire's decision they can challenge it to the chair umpire, at which point the chair umpire uses electronic review to make a final decision about the call.

If a player's challenge is upheld, then the player gets to keep the point which was lost and his challenge, which is referred to as an *overturn*. For example, if the current score is 30-15, and if the server won the current play the score becomes 40-15 and if it is challenged correctly by the opponent the

score changes to 30-30. We can see that the initial and final scores are not the same in this situation. If the player's challenge is not upheld, then he loses one of the limited opportunities to challenge the line umpire's call. There might be some situations where a player does not return a ball because a linesman called out. This situation may be challenged by a player and if the challenge is upheld, the point may be replayed. If this type of point is deemed unreturnable by the umpire, then an overturn occurs.

If a player's first serve is an apparent ace and it is challenged correctly by his opponent, then the player gets a second serve. For example if the current score is 0-0, after an ace on the first serve score becomes 15-0 and if it is challenged correctly by the opponent the score goes back to 0-0. In this situation we cannot call it as an overturn because the opponent did not gain a point even after challenging correctly. Let us refer to this situation as *second serve replay*. Thus, there are three types of challenge results: those that result in an overturned point, those that result in a replay and those that result in a second serve. Second serve replays are not considered in our model as it requires keeping track of aces, resulting in a huge state space.

A player will have a maximum of three chances to incorrectly challenge, under the advantage set system. If both the players win six games each, the challenge counter is reset giving them three incorrect challanges each, which could be used anytime during the next 12 games. Resetting is repeated after every 12 games.

A player can challenge only on either a point-ending shot or when he

stops playing a point. Even if he returns the ball he must stop immediately in order to challenge. If the chair umpire feels that the player did not challenge in time, he might refuse the appeal for electronic review. In some situations if the line umpire is not able to make a decision, an electronic review may be invoked by the chair umpire.

With this background about the point challenge system, some interesting challenge statistics from some of the recent major tennis tournaments are included in Tables 1.1 and 1.2. Looking at the men's statistics, the average number of challenges per match is 6 approximately. The minimum number of points required to complete a match is 72 (minimum of 4 points to win a game, 6 games to win a set and 3 sets to win a match, which results in a total of 72 points). There is no absolute maximum on the number of points played in a match because there is no upper limit on the number of points played in any given game. Suppose that a typical match consists of 10 points per game, 10 games per set and 4 sets per match. Under these assumptions there are about 400 points per match. So approximately 1.5% of the total points played were being challenged. Out of these challenges only 30% resulted in overturned calls.

Table 1.1: Singles challenge summary US Open 2009

	Men's	Women's
Total Number of Challenges	360	220
Number of Correct Challenges	105	59
Number of Incorrect Challenges	255	161
Percentage Overturned	29.76%	26.82%
Avg. Challenges per Match	6.32	4.00

Table 1.2: Singles challenge summary Wimbledon 2010

	Men's	Women's
Total Number of Challenges	260	140
Number of Correct Challenges	92	44
Number of Incorrect Challenges	168	96
Percentage Overturned	35.38%	31.43%
Avg. Challenges per Match	5.65	4.24

Chapter 2

Methodology

The basic aim of this work is to get insight on when to use the limited opportunity to challenge an umpire's call with an objective of improving the overall probability of winning a tennis match. As discussed in the point challenge system earlier, we are aware that these limited opportunities to challenge are not carried over from one set to another. Thus, in order to model challenges in a tennis match, it is sufficient to model a single set. Because of the nested structure of scoring in tennis, the best way to come up with a model for the whole set is to develop a model for a single game and later extend it to include the whole set.

So, in this work we model a single game in a tennis match. Some important assumptions were made to reduce the complexity involved and for the ease of analysis. One such assumption is of considering the game to be a normal one, where only one player serves all the time. A tie-breaker game involves complex serve change rules and and hence is not considered for the time being. Another assumption is that the events in game are independent and identically distributed (i.i.d). This means that the result of a point has no bearing on the result of the subsequent played points. Also a player's

probability of winning a point is considered to be constant throughout the game. This i.i.d assumption is quite severe but past work suggests that it is adequate for analysis [1]. From here on by *challenges* we mean the number of opportunities to make incorrect challenges to an umpire's call.

In order to define the MDP model we need to explain the four basic elements of it: the state space, actions, dynamics (or probability transition matrices) and the reward function.

2.1 State space

While defining the state we need to consider different variables like the current situation of the game (score), the number of challenges left, the result of the current played point, the player's perception of the challengability on the current played point and whether the result of the successful challenge is an overturn or a replay. Each of these variables has an influence on whether to challenge or not in a particular situation. Our states should include all these variables in order to reflect the real situation in a tennis match. Also, we do not consider the "strategic" use of challenges as applied to the other player's remaining challenges. In other words, our state does not include the opposing player's remaining challenges.

The current score is one of the most important components of our state space. As we are modeling a single game, we create a representation of all the possible scores into a two digit number. The score of a player is converted into a single digit as represented below:

- Love $\rightarrow$ 0
- 15 $\rightarrow$ 1
- 30 $\rightarrow$ 2
- 40 $\rightarrow$ 3
- Game $\rightarrow$ 4.

With this conversion the possible scores in a game can be divided into two sets. The first set has the following elements 00 , 01, 02, 03, 11, 12, 13, 22, 23, 10, 20, 30, 21, 31 and 32, where the first digit represents the server's points and the second digit represents the other player's points. For example the number 23 represents the score $30 - 40$. The elements of the second set are 33, 34, 43, 04 and 40, which in turn have meanings as follows:

- 33 $\rightarrow$ deuce
- 43 $\rightarrow$ advantage in
- 34 $\rightarrow$ advantage out
- 04 $\rightarrow$ game lost
- 40 $\rightarrow$ game won.

A player's probability of winning a point p has a significant effect on the decision making process. In order to estimate this value for a contest

between two players, we need some statistics of previous encounters of these two players. However, we can assume a value of say 0.6, and vary it if necessary for the sake of computational studies.

Another component of our state space is the result of the current played point. This component can take three values depending on whether you lost the point or you won the point and the third value for indicating that the game is over. In our model this component is described using the following notation:

- $1 \rightarrow$ indicates that server won the current played point
- $2 \rightarrow$ indicates that server lost the current played point
- $3 \rightarrow$ indicates that the game is over.

Another component of our state is the number of challenges left for the player. Assuming that there are a total of q challenges per game, the possible values this component can take are 0, 1, 2, ..., q. The meaning of different values of this component is as follows:

- $0 \rightarrow$ The player has no challenges left.
- $1 \rightarrow$ The player has one challenge left.

$\vdots$

- $q \rightarrow$ The player has q challenges left.

The perception of winning a challenge is an interesting variable that has been included in our state. If the ball is called out and it appears to be, say, 3 feet out to the player, then in this case it is evident that the umpire has made the correct call. In such a case there is no point in challenging that call. Some players might challenge even in such situations when, for example, he has challenges left and is about to lose the game. In other situations the player may have a doubt about the umpire's call and it will be useful to know what action he takes depending on how sure he is about winning the challenge. In order to map the player's perception with the action he takes, this component is included which can take five possible values, whose notation and meaning is as follows:

- $1 \rightarrow$ The player is $s_1\%$ sure that he is going to win the challenge.
- $2 \rightarrow$ The player is $s_2\%$ sure that he is going to win the challenge.
- $3 \rightarrow$ The player is $s_3\%$ sure that he is going to win the challenge.
- $4 \rightarrow$ The player is $s_4\%$ sure that he is going to win the challenge.
- $5 \rightarrow$ The player's perception does not matter.

The value of s_1 will be 0 typically and the values of s_2, s_3, s_4 will be increasing in order. The probabilities and values of the perception states are parameters which we adjust in our computational investigations.

The last component of the state is to indicate whether the point will be overturned or replayed in case the call is challenged correctly. This component can take three values whose notation and meaning is given as follows:

- $1 \rightarrow$ The point will be replayed, when the call is challenged correctly.
- $2 \rightarrow$ The point will be overturned, when the call is challenged correctly.
- $3 \rightarrow$ It does not matter whether there is a replay or not.

Our state turns out to be a five dimensional one with each dimension representing one of the components discussed earlier. We can convert our five dimensional state into a six digit number "abcdef" for the sake of computational benefit in generating the transition probability matrices and for developing the linear program. In the number "abcdef", 'a' indicates result of the current played point, 'bc' indicates the current score, 'd' represents the number of challenges left, 'e' indicates player's perception of winning a challenge, 'f' indicates whether the point is replayed or not in case of a successful challenge.

In the five dimensional state, each component in each dimension can assume some values from a set of possibles values which were discussed earlier. If each and every combination is allowed, this should result in a total number of $3 \cdot 20 \cdot (q+1) \cdot 5 \cdot 3$ i.e., $900 \cdot (q+1)$ states, but most of these states are not meaningful. The total number of allowed states are illustrated below

a	$\cdot$	bc	$\cdot$	d	$\cdot$	e	$\cdot$	f
[3]	$\cdot$	$[40, 04]$	$\cdot$	$[0, 1, \ldots, q]$	$\cdot$	[5]	$\cdot$	[3]
[1]	$\cdot$	[scores except $40, 04$]	$\cdot$	$[0, 1, \ldots, q]$	$\cdot$	[5]	$\cdot$	[3]
[2]	$\cdot$	[scores except $40, 04$]	$\cdot$	$[1, \ldots, q]$	$\cdot$	$[1, 2, 3, 4]$	$\cdot$	$[1, 2]$
[2]	$\cdot$	[scores except $40, 04$]	$\cdot$	[0]	$\cdot$	[5]	$\cdot$	[3],

which results in a total of $164q + 38$ states. The higher the number of challenges per game (q) available to a player, the higher the number of states and hence the bigger the size of transition probability matrix. The size of transition probability matrix also depends on the total number of actions available, which in our model is just two. So if there are a total of b possible states, the size of transition probability matrix would be $b \times 2b$. These size issues are further illustrated in Table 2.1.

Table 2.1: Number of challenges and size of transition probability matrix

Number of challenges per game	Size of transition probability matrix
1	202×404
2	370×740
3	538×1076
$\vdots$	$\vdots$
q	$(168q + 34) \times (336q + 68)$

Each state represents a situation in a tennis match depending on the value each component takes. For example, the state 233132 says that the game is at deuce, the player lost the current played point, he is left with a challenge, he is s_3% sure that he is going to win if he challenges and the point does not get replayed if he challenges correctly.

2.2 Actions

There are two possible actions that can be chosen in each state. They are:

- c $\rightarrow$ the player does challenge
- nc $\rightarrow$ the player does not challenge.

In some states, both actions are available to the player. In other states, only the *do not challenge* action is available, e.g., when the player has no challenges left. We have to restrict the availability of actions depending on the state we are in. In our case, if the first digit of the state is either 1 or 3, or if the fifth digit is 1, the only action available will be *nc*. In the rest of the states the player can take either of the available actions.

2.3 Transition probability matrix

In order to generate the transition probability $p_{ij}(u)$ of ending up in state j when you apply an action u in the state i, we need to know both the states as well as the action taken. Apart from this it is important here to define some variables as follows, which are necessary for the generation of transition probability matrices. We now describe the basic parameters which serve as inputs to the model.

The first is p, the probability that the player who is the decision maker, wins any given point (recall our i.i.d. assumption regarding individual points).

The next parameter is r, the probability that a point is replayed given that there is a successful challenge. Finally, for each perceptual state s_i, p_i is the probability of being in that state after a given play. Note that $p_1+p_2+p_3+p_4 = 1$.

For example suppose a player issues a challenge in the initial state 212122 and ends up in state 222111. The transition probability involved in this case would be $[1-p] \cdot s_2 \cdot p_1 \cdot [1-r]$, because if we look at the initial state it is a situation where the player lost the current played point when the score is 12 and ended up in a state where the score is 22. The player actually gained a point after losing the point which means that he challenged the call correctly (the probability of this happening is s_2). The term p_1 corresponds to ending up in a state where the component of players perception has a value 1. Similarly the term $1-r$ says that the player ended up in a state where there will be no replay in case of a correct challenge.

A C++ code has been developed to generate the transition probability matrices and it has been included in Appendix A for reference.

2.4 Rewards

In our model, the states can be classified into either recurrent or transient as shown in Figure 2.1. If the player enters a state where he wins the game, he is not allowed to challenge and will eternally stay in that state. The situation is similar when he loses the game. There are a set of intermediate states which are transient, where the player has a set of actions to take de-

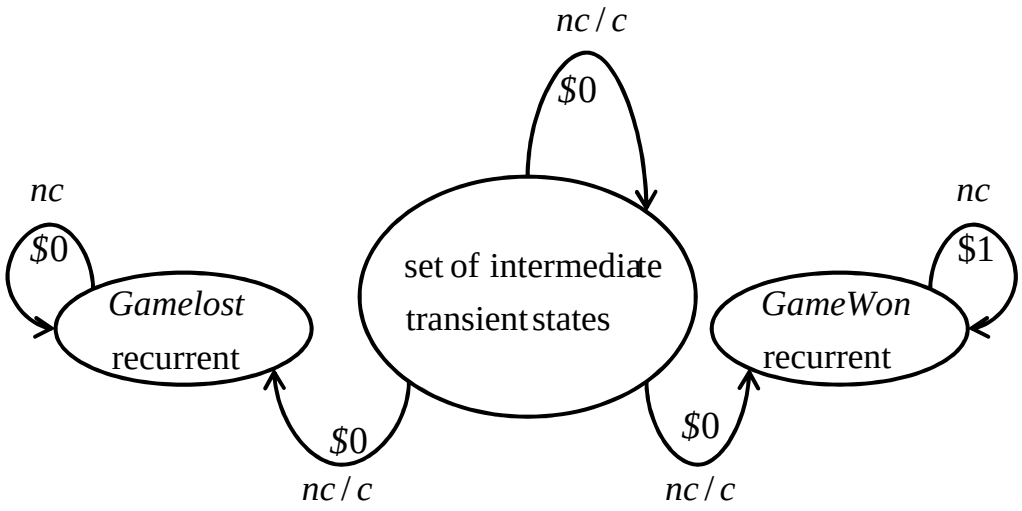

Figure 2.1: State transitions and rewards

pending on which state he is in. The player might spend some time in these states but eventually will end up in one of the recurrent states when he wins or loses the game. A reward of \$1 is earned each time the player ends up in a winning state. So, in the long run, if a player wins, his average reward would be \$1 and if he loses his reward would be \$0. There is no reward for any other transitions. Thus, maximizing the long-run average reward in this model is equivalent to maximizing the probability of winning the game.

In the Figure 2.1 we can see that there are two recurrent classes along with a set of transient states. This implies that the model falls into the category of infinite horizon multi-chain average cost Markov decision processes.

2.5 Linear Programming formulation for the multi-chain case

We have discussed the linear programming approach to obtain the optimal policy in the multi-chain case earlier in section 1.1.4 and we are adapting this approach for our current problem. A GAMS model, which appears in Appendix B, has been developed to solve this linear program. This GAMS code takes the transition probability matrices as input in the form of csv (comma separated value) file, that was generated with the help of the C++ code.

Chapter 3

Computational experiments and results

3.1 Overview

The purpose of these experiments is to the explore the effect on the optimal policy of variables like the player's probability of winning a point p, the player's perception of challengability of a call (s_1, s_2, s_3, s_4) and the proportion of time a player ends up in a state where the specific perception of winning a challenge (p_1, p_2, p_3, p_4). We can vary the values of these variables in the C++ program and obtain the corresponding transition probability matrix which in turn is used as an input to the GAMS model that gives us the optimal policy. All the computational experiments were done considering that only one challenge is available to the player per game. This results in a total of 202 possible states and only one action nc is available in some of them. These are the states where the first digit in the state is either 3 or 1, which correspond to states in which the player either has no challenges left, or in which the game is over. Other states where the first digit in the state is 2 and the fifth digit is 1 also have only one action nc available to take and these correspond to the states in which the player's perceived probability of winning the challenge is 0. Note that a challenge could be included as an available action in such states. However, in most such states these actions would be suboptimal. In

some states (such as those where a game will be lost anyway) the challenge or no challenge option leads to the same result. Thus eliminating the actions in such states simplifies the model without eliminating optimal policies. In order to present the tables in a compact form, all the states where the only action available is *nc* for any value of the parameters are not included. Also states are grouped together based on the current score in the tables. In the following sections, results of three different experiments are presented and discussed.

3.2 Effect of varying p_1

In this experiment the proportion of time that a player ends up in a state where his perception of winning a challenge is 0%, i.e., p_1 is varied from 0.1 to 0.9. Our intuition, and some of the match statistics presented early imply that p_1 should be quite high in reality (i.e., above 0.95). However, to get insight into the model dynamics, we study the system for a range of p_1 values. The remaining time is divided equally between states where the perception of winning a challenge is s_2 or s_3 or s_4 percent, i.e, $p_2 = p_3 = p_4 = (1 - p_1)/3$. Initially, we set the four perception levels to be $s_1 = 0, s_2 = 25, s_3 = 50$ and $s_4 = 75$. In these experiments we also fix $p = 0.6$ and $r = 0.6$. The results are shown in the Table 3.1 below. The value of p_1 is varied across the column and each row corresponds to a different state. The values in the table show the optimal action for each combination of state and p_1.

Table 3.1: Effect of p_1

		p_1								
States	**Score, Perception, Result of successful challenge**	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
200121	0-0, 25%, replay	nc	nc	nc	nc	nc	nc	nc	nc	c
200122	0-0, 25%, no replay	nc	nc	nc	nc	nc	nc	nc	c	c
200131	0-0, 50%, replay	nc	nc	nc	nc	nc	c	c	c	c
200132	0-0, 50%, no replay	c	c	c	c	c	c	c	c	c
200141	0-0, 75%, replay	c	c	c	c	c	c	c	c	c
200142	0-0, 75%, no replay	c	c	c	c	c	c	c	c	c
201121	0-15, 25%, replay	nc	nc	nc	nc	nc	nc	nc	c	c
201122	0-15, 25%, no replay	nc	nc	nc	nc	nc	c	c	c	c
201131	0-15, 50%, replay	nc	nc	nc	c	c	c	c	c	c
201132	0-15, 50%, no replay	c	c	c	c	c	c	c	c	c
201141	0-15, 75%, replay	c	c	c	c	c	c	c	c	c
201142	0-15, 75%, no replay	c	c	c	c	c	c	c	c	c
202121	0-30, 25%, replay	nc	nc	nc	nc	nc	c	c	c	c
202122	0-30, 25%, no replay	nc	nc	nc	c	c	c	c	c	c
202131	0-30, 50%, replay	c	c	c	c	c	c	c	c	c
202132	0-30, 50%, no replay	c	c	c	c	c	c	c	c	c
202141	0-30, 75%, replay	c	c	c	c	c	c	c	c	c
202142	0-30, 75%, no replay	c	c	c	c	c	c	c	c	c
203121	0-40, 25%, replay	c	c	c	c	c	c	c	c	c
203122	0-40, 25%, no replay	c	c	c	c	c	c	c	c	c
203131	0-40, 50%, replay	c	c	c	c	c	c	c	c	c
203132	0-40, 50%, no replay	c	c	c	c	c	c	c	c	c
203141	0-40, 75%, replay	c	c	c	c	c	c	c	c	c
203142	0-40, 75%, no replay	c	c	c	c	c	c	c	c	c
210121	15-0, 25%, replay	nc	nc	nc	nc	nc	nc	nc	nc	c
210122	15-0, 25%, no replay	nc	nc	nc	nc	nc	nc	nc	c	c
210131	15-0, 50%, replay	nc	nc	nc	nc	nc	c	c	c	c
210132	15-0, 50%, no replay	nc	c	c	c	c	c	c	c	c

Continued on Next Page...

Table 3.1 – Continued

		p_1								
States	**Score, Perception, Result of successful challenge**	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
210141	15-0, 75%, replay	c	c	c	c	c	c	c	c	c
210142	15-0, 75%, no replay	c	c	c	c	c	c	c	c	c
211121	15-15, 25%, replay	nc	nc	nc	nc	nc	nc	nc	c	c
211122	15-15, 25%, no replay	nc	nc	nc	nc	nc	nc	c	c	c
211131	15-15, 50%, replay	nc	nc	nc	c	c	c	c	c	c
211132	15-15, 50%, no replay	c	c	c	c	c	c	c	c	c
211141	15-15, 75%, replay	c	c	c	c	c	c	c	c	c
211142	15-15, 75%, no replay	c	c	c	c	c	c	c	c	c
212121	15-30, 25%, replay	nc	nc	nc	nc	nc	nc	c	c	c
212122	15-30, 25%, no replay	nc	nc	nc	c	c	c	c	c	c
212131	15-30, 50%, replay	c	c	c	c	c	c	c	c	c
212132	15-30, 50%, no replay	c	c	c	c	c	c	c	c	c
212141	15-30, 75%, replay	c	c	c	c	c	c	c	c	c
212142	15-30, 75%, no replay	c	c	c	c	c	c	c	c	c
213121	15-40, 25%, replay	c	c	c	c	c	c	c	c	c
213122	15-40, 25%, no replay	c	c	c	c	c	c	c	c	c
213131	15-40, 50%, replay	c	c	c	c	c	c	c	c	c
213132	15-40, 50%, no replay	c	c	c	c	c	c	c	c	c
213141	15-40, 75%, replay	c	c	c	c	c	c	c	c	c
213142	15-40, 75%, no replay	c	c	c	c	c	c	c	c	c
220121	30-0, 25%, replay	nc	nc	nc	nc	nc	nc	nc	nc	c
220122	30-0, 25%, no replay	nc	nc	nc	nc	nc	nc	nc	c	c
220131	30-0, 50%, replay	nc	nc	nc	nc	nc	c	c	c	c
220132	30-0, 50%, no replay	nc	c	c	c	c	c	c	c	c
220141	30-0, 75%, replay	c	c	c	c	c	c	c	c	c
220142	30-0, 75%, no replay	c	c	c	c	c	c	c	c	c
221121	30-15, 25%, replay	nc	nc	nc	nc	nc	nc	nc	nc	c
221122	30-15, 25%, no replay	nc	nc	nc	nc	nc	nc	c	c	c

Continued on Next Page...

Table 3.1 – Continued

		p_1								
States	**Score, Perception, Result of successful challenge**	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
221131	30-15, 50%, replay	nc	nc	nc	nc	c	c	c	c	c
221132	30-15, 50%, no replay	c	c	c	c	c	c	c	c	c
221141	30-15, 75%, replay	c	c	c	c	c	c	c	c	c
221142	30-15, 75%, no replay	c	c	c	c	c	c	c	c	c
222121	30-30, 25%, replay	nc	nc	nc	nc	nc	nc	c	c	c
222122	30-30, 25%, no replay	nc	nc	nc	nc	c	c	c	c	c
222131	30-30, 50%, replay	nc	c	c	c	c	c	c	c	c
222132	30-30, 50%, no replay	c	c	c	c	c	c	c	c	c
222141	30-30, 75%, replay	c	c	c	c	c	c	c	c	c
222142	30-30, 75%, no replay	c	c	c	c	c	c	c	c	c
223121	30-40, 25%, replay	c	c	c	c	c	c	c	c	c
223122	30-40, 25%, no replay	c	c	c	c	c	c	c	c	c
223131	30-40, 50%, replay	c	c	c	c	c	c	c	c	c
223132	30-40, 50%, no replay	c	c	c	c	c	c	c	c	c
223141	30-40, 75%, replay	c	c	c	c	c	c	c	c	c
223142	30-40, 75%, no replay	c	c	c	c	c	c	c	c	c
230121	40-0, 25%, replay	nc	nc	nc	nc	nc	nc	nc	nc	c
230122	40-0, 25%, no replay	nc	nc	nc	nc	nc	nc	nc	c	c
230131	40-0, 50%, replay	nc	nc	nc	nc	nc	nc	c	c	c
230132	40-0, 50%, no replay	nc	nc	c	c	c	c	c	c	c
230141	40-0, 75%, replay	c	c	c	c	c	c	c	c	c
230142	40-0, 75%, no replay	c	c	c	c	c	c	c	c	c
231121	40-15, 25%, replay	nc	nc	nc	nc	nc	nc	nc	nc	c
231122	40-15, 25%, no replay	nc	nc	nc	nc	nc	nc	c	c	c
231131	40-15, 50%, replay	nc	nc	nc	nc	c	c	c	c	c
231132	40-15, 50%, no replay	c	c	c	c	c	c	c	c	c
231141	40-15, 75%, replay	c	c	c	c	c	c	c	c	c
231142	40-15, 75%, no replay	c	c	c	c	c	c	c	c	c

Continued on Next Page...

Table 3.1 – Continued

		p_1								
States	**Score, Perception, Result of successful challenge**	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
232121	40-30, 25%, replay	nc	nc	nc	nc	nc	nc	nc	c	c
232122	40-30, 25%, no replay	nc	nc	nc	nc	nc	c	c	c	c
232131	40-30, 50%, replay	nc	nc	c	c	c	c	c	c	c
232132	40-30, 50%, no replay	c	c	c	c	c	c	c	c	c
232141	40-30, 75%, replay	c	c	c	c	c	c	c	c	c
232142	40-30, 75%, no replay	c	c	c	c	c	c	c	c	c
233121	deuce, 25%, replay	nc	nc	nc	nc	nc	nc	c	c	c
233122	deuce, 25%, no replay	nc	nc	nc	nc	c	c	c	c	c
233131	deuce, 50%, replay	nc	c	c	c	c	c	c	c	c
233132	deuce, 50%, no replay	c	c	c	c	c	c	c	c	c
233141	deuce, 75%, replay	c	c	c	c	c	c	c	c	c
233142	deuce, 75%, no replay	c	c	c	c	c	c	c	c	c
234121	ad out, 25%, replay	c	c	c	c	c	c	c	c	c
234122	ad out, 25%, no replay	c	c	c	c	c	c	c	c	c
234131	ad out, 50%, replay	c	c	c	c	c	c	c	c	c
234132	ad out, 50%, no replay	c	c	c	c	c	c	c	c	c
234141	ad out, 75%, replay	c	c	c	c	c	c	c	c	c
234142	ad out, 75%, no replay	c	c	c	c	c	c	c	c	c
243121	ad in, 25%, replay	nc	nc	nc	nc	nc	nc	nc	c	c
243122	ad in, 25%, no replay	nc	nc	nc	nc	nc	c	c	c	c
243131	ad in, 50%, replay	nc	nc	c	c	c	c	c	c	c
243132	ad in, 50%, no replay	c	c	c	c	c	c	c	c	c
243141	ad in, 75%, replay	c	c	c	c	c	c	c	c	c
243142	ad in, 75%, no replay	c	c	c	c	c	c	c	c	c

First of all, Table 3.1 confirms our intuition that it is always optimal to challenge a call whenever the player has a challenge left and when he going to lose the game by losing the current played point i.e., when the current score

is either 03, 13, 23 or 34. This is true for any value of p_1 and for any value of the player's perception. In a particular state, as the value of p_1 increases we can see a change in the optimal action taken. For example in the state 243122 (row highlighted in green in Table 3.1) it is optimal to not challenge until the value of p_1 becomes 0.6 and then it is optimal to challenge. This means that as the player's probability of seeing a challengable point is low he challenges and when it is above a threshold value he does not challenge. For a particular group of states which have the same score we see that the higher the perception of winning, the better it is to challenge. If we look at the last column in the results table where the value of p_1 is 0.9, the optimal action is to challenge no matter what state the player is in. For this value of p_1 the proportion of time the player ends up in a state when he has some positive perception of winning a challenge is very small and so it is optimal to use this challenge whenever he is in such a state.

3.3 Effect of varying p

The purpose of this set of experiments is to see how a player's strength with respect to his opponent affects the optimal action he takes in different situations. In this experiment the player's probability of winning a point is varied from 0.1 to 0.9. Each player has four levels of perception and they are $s_1 = 0, s_2 = 25, s_3 = 50$ and $s_4 = 75$. We again fix $r = 0.6$. Three different values for p_1 i.e., 0.1, 0.5 and 0.9 are used while keeping all other variables the same. The results corresponding to each value of p_1 are shown

in Tables 3.4, 3.3 and 3.2. For each value of p_1, the values of p_2, p_3, p_4 are obtained using the following expression $p_2 = p_3 = p_4 = (1 - p_1)/3$. The value of p is varied across the columns and each row corresponds to a different state. The values in the table show the optimal action for each combination of state and p.

3.3.1 Case 1: Value of p_1 is 0.9

The results shown in Table 3.2 below shows that the optimal action in this case is to challenge irrespective of the state the player is in and the value of p, except for in a very few states like 200121, 210121, 220121 and 230121, when the value of p is 0.1 (highlighted in green in the Table 3.2). Looking at these highighted states we can say that when the proportion of time the player ends up in a state where he has a positive perception of winning a challenge is very low, he will challenge whenever he has a chance except in a very few situations. Such situations are when his probability of winning a point is very low (0.1), his perception of winning a challenge is 25%, the scores are 00, 10, 20 or 30 and when the point will be replayed if it is challenged correctly.

Table 3.2: Effect of varying p when p_1 is 0.9

		p								
States	**Score, Perception, Result of successful challenge**	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
200121	0-0, 25%, replay	nc	c	c	c	c	c	c	c	c
200122	0-0, 25%, no replay	c	c	c	c	c	c	c	c	c
200131	0-0, 50%, replay	c	c	c	c	c	c	c	c	c
200132	0-0, 50%, no replay	c	c	c	c	c	c	c	c	c

Continued on Next Page...

Table 3.2 – Continued

		p								
States	**Score, Perception, Result of successful challenge**	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
200141	0-0, 75%, replay	c	c	c	c	c	c	c	c	c
200142	0-0, 75%, no replay	c	c	c	c	c	c	c	c	c
201121	0-15, 25%, replay	c	c	c	c	c	c	c	c	c
201122	0-15, 25%, no replay	c	c	c	c	c	c	c	c	c
201131	0-15, 50%, replay	c	c	c	c	c	c	c	c	c
201132	0-15, 50%, no replay	c	c	c	c	c	c	c	c	c
201141	0-15, 75%, replay	c	c	c	c	c	c	c	c	c
201142	0-15, 75%, no replay	c	c	c	c	c	c	c	c	c
202121	0-30, 25%, replay	c	c	c	c	c	c	c	c	c
202122	0-30, 25%, no replay	c	c	c	c	c	c	c	c	c
202131	0-30, 50%, replay	c	c	c	c	c	c	c	c	c
202132	0-30, 50%, no replay	c	c	c	c	c	c	c	c	c
202141	0-30, 75%, replay	c	c	c	c	c	c	c	c	c
202142	0-30, 75%, no replay	c	c	c	c	c	c	c	c	c
203121	0-40, 25%, replay	c	c	c	c	c	c	c	c	c
203122	0-40, 25%, no replay	c	c	c	c	c	c	c	c	c
203131	0-40, 50%, replay	c	c	c	c	c	c	c	c	c
203132	0-40, 50%, no replay	c	c	c	c	c	c	c	c	c
203141	0-40, 75%, replay	c	c	c	c	c	c	c	c	c
203142	0-40, 75%, no replay	c	c	c	c	c	c	c	c	c
210121	15-0, 25%, replay	nc	c	c	c	c	c	c	c	c
210122	15-0, 25%, no replay	c	c	c	c	c	c	c	c	c
210131	15-0, 50%, replay	c	c	c	c	c	c	c	c	c
210132	15-0, 50%, no replay	c	c	c	c	c	c	c	c	c
210141	15-0, 75%, replay	c	c	c	c	c	c	c	c	c
210142	15-0, 75%, no replay	c	c	c	c	c	c	c	c	c
211121	15-15, 25%, replay	c	c	c	c	c	c	c	c	c
211122	15-15, 25%, no replay	c	c	c	c	c	c	c	c	c

Continued on Next Page. . .

Table 3.2 – Continued

		p								
States	**Score, Perception, Result of successful challenge**	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
211131	15-15, 50%, replay	c	c	c	c	c	c	c	c	c
211132	15-15, 50%, no replay	c	c	c	c	c	c	c	c	c
211141	15-15, 75%, replay	c	c	c	c	c	c	c	c	c
211142	15-15, 75%, no replay	c	c	c	c	c	c	c	c	c
212121	15-30, 25%, replay	c	c	c	c	c	c	c	c	c
212122	15-30, 25%, no replay	c	c	c	c	c	c	c	c	c
212131	15-30, 50%, replay	c	c	c	c	c	c	c	c	c
212132	15-30, 50%, no replay	c	c	c	c	c	c	c	c	c
212141	15-30, 75%, replay	c	c	c	c	c	c	c	c	c
212142	15-30, 75%, no replay	c	c	c	c	c	c	c	c	c
213121	15-40, 25%, replay	c	c	c	c	c	c	c	c	c
213122	15-40, 25%, no replay	c	c	c	c	c	c	c	c	c
213131	15-40, 50%, replay	c	c	c	c	c	c	c	c	c
213132	15-40, 50%, no replay	c	c	c	c	c	c	c	c	c
213141	15-40, 75%, replay	c	c	c	c	c	c	c	c	c
213142	15-40, 75%, no replay	c	c	c	c	c	c	c	c	c
220121	30-0, 25%, replay	nc	c	c	c	c	c	c	c	c
220122	30-0, 25%, no replay	c	c	c	c	c	c	c	c	c
220131	30-0, 50%, replay	c	c	c	c	c	c	c	c	c
220132	30-0, 50%, no replay	c	c	c	c	c	c	c	c	c
220141	30-0, 75%, replay	c	c	c	c	c	c	c	c	c
220142	30-0, 75%, no replay	c	c	c	c	c	c	c	c	c
221121	30-15, 25%, replay	c	c	c	c	c	c	c	c	c
221122	30-15, 25%, no replay	c	c	c	c	c	c	c	c	c
221131	30-15, 50%, replay	c	c	c	c	c	c	c	c	c
221132	30-15, 50%, no replay	c	c	c	c	c	c	c	c	c
221141	30-15, 75%, replay	c	c	c	c	c	c	c	c	c
221142	30-15, 75%, no replay	c	c	c	c	c	c	c	c	c

Continued on Next Page...

Table 3.2 – Continued

		p								
States	**Score, Perception, Result of successful challenge**	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
222121	30-30, 25%, replay	c	c	c	c	c	c	c	c	c
222122	30-30, 25%, no replay	c	c	c	c	c	c	c	c	c
222131	30-30, 50%, replay	c	c	c	c	c	c	c	c	c
222132	30-30, 50%, no replay	c	c	c	c	c	c	c	c	c
222141	30-30, 75%, replay	c	c	c	c	c	c	c	c	c
222142	30-30, 75%, no replay	c	c	c	c	c	c	c	c	c
223121	30-40, 25%, replay	c	c	c	c	c	c	c	c	c
223122	30-40, 25%, no replay	c	c	c	c	c	c	c	c	c
223131	30-40, 50%, replay	c	c	c	c	c	c	c	c	c
223132	30-40, 50%, no replay	c	c	c	c	c	c	c	c	c
223141	30-40, 75%, replay	c	c	c	c	c	c	c	c	c
223142	30-40, 75%, no replay	c	c	c	c	c	c	c	c	c
230121	40-0, 25%, replay	nc	c	c	c	c	c	c	c	c
230122	40-0, 25%, no replay	c	c	c	c	c	c	c	c	c
230131	40-0, 50%, replay	c	c	c	c	c	c	c	c	c
230132	40-0, 50%, no replay	c	c	c	c	c	c	c	c	c
230141	40-0, 75%, replay	c	c	c	c	c	c	c	c	c
230142	40-0, 75%, no replay	c	c	c	c	c	c	c	c	c
231121	40-15, 25%, replay	c	c	c	c	c	c	c	c	c
231122	40-15, 25%, no replay	c	c	c	c	c	c	c	c	c
231131	40-15, 50%, replay	c	c	c	c	c	c	c	c	c
231132	40-15, 50%, no replay	c	c	c	c	c	c	c	c	c
231141	40-15, 75%, replay	c	c	c	c	c	c	c	c	c
231142	40-15, 75%, no replay	c	c	c	c	c	c	c	c	c
232121	40-30, 25%, replay	c	c	c	c	c	c	c	c	c
232122	40-30, 25%, no replay	c	c	c	c	c	c	c	c	c
232131	40-30, 50%, replay	c	c	c	c	c	c	c	c	c
232132	40-30, 50%, no replay	c	c	c	c	c	c	c	c	c
232141	40-30, 75%, replay	c	c	c	c	c	c	c	c	c

Continued on Next Page. . .

Table 3.2 – Continued

		p								
States	**Score, Perception, Result of successful challenge**	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
232142	40-30, 75%, no replay	c	c	c	c	c	c	c	c	c
233121	deuce, 25%, replay	c	c	c	c	c	c	c	c	c
233122	deuce, 25%, no replay	c	c	c	c	c	c	c	c	c
233131	deuce, 50%, replay	c	c	c	c	c	c	c	c	c
233132	deuce, 50%, no replay	c	c	c	c	c	c	c	c	c
233141	deuce, 75%, replay	c	c	c	c	c	c	c	c	c
233142	deuce, 75%, no replay	c	c	c	c	c	c	c	c	c
234121	ad out, 25%, replay	c	c	c	c	c	c	c	c	c
234122	ad out, 25%, no replay	c	c	c	c	c	c	c	c	c
234131	ad out, 50%, replay	c	c	c	c	c	c	c	c	c
234132	ad out, 50%, no replay	c	c	c	c	c	c	c	c	c
234141	ad out, 75%, replay	c	c	c	c	c	c	c	c	c
234142	ad out, 75%, no replay	c	c	c	c	c	c	c	c	c
243121	ad in, 25%, replay	c	c	c	c	c	c	c	c	c
243122	ad in, 25%, no replay	c	c	c	c	c	c	c	c	c
243131	ad in, 50%, replay	c	c	c	c	c	c	c	c	c
243132	ad in, 50%, no replay	c	c	c	c	c	c	c	c	c
243141	ad in, 75%, replay	c	c	c	c	c	c	c	c	c
243142	ad in, 75%, no replay	c	c	c	c	c	c	c	c	c

3.3.2 Case 2: Value of p_1 is 0.5

In Table 3.3 below, states are grouped based on score. In each group as we move down the rows, we can see that optimal action changes from *nc* to *c*. This is as expected because the higher your perceived probability of winning a challenge is, the better it is to challenge. Also in each row as we move from

left to right the optimal action changes from c to challenge nc except in some states. This says that as a player becomes stronger (relative to his opponent), it becomes unnecessary for him to use his challenges. But in some states like 200131 for example (such states were shown by rows highlighted in green in Table 3.3), as we move from left to right we can see that the optimal action is c initially and then nc and then again shifts to c. This says that the player has to challenge if he is either too weak or too strong. This makes sense because if he is too weak, challenging is the only way he could win a point and if he is too strong, losing challenges does not make as much difference to him.

Table 3.3: Effect of varying p when p_1 is 0.5

		p								
States	**Score, Perception, Result of successful challenge**	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
200121	0-0, 25%, replay	nc	nc	nc	nc	nc	nc	nc	nc	nc
200122	0-0, 50%, no replay	c	c	nc	nc	nc	nc	nc	nc	nc
200131	0-0, 25%, replay	c	nc	nc	nc	nc	c	c	c	c
200132	0-0, 50%, replay	c	c	c	c	c	c	c	c	c
200141	0-0, 75%, replay	c	c	c	c	c	c	c	c	c
200142	0-0, 75%, no replay	c	c	c	c	c	c	c	c	c
201121	0-15, 25%, replay	nc	nc	nc	nc	nc	nc	nc	nc	nc
201122	0-15, 25%, no replay	c	c	c	c	nc	nc	nc	nc	nc
201131	0-15, 50%, replay	c	c	c	c	c	c	c	c	c
201132	0-15, 50%, no replay	c	c	c	c	c	c	c	c	c
201141	0-15, 75%, replay	c	c	c	c	c	c	c	c	c
201142	0-15, 75%, no replay	c	c	c	c	c	c	c	c	c
202121	0-30, 25%, replay	c	c	nc	nc	nc	nc	nc	nc	nc
202122	0-30, 25%, no replay	c	c	c	c	c	c	c	c	c
202131	0-30, 50%, replay	c	c	c	c	c	c	c	c	c

Continued on Next Page...

Table 3.3 – Continued

		p								
States	**Score, Perception, Result of successful challenge**	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
202132	0-30, 50%, no replay	c	c	c	c	c	c	c	c	c
202141	0-30, 75%, replay	c	c	c	c	c	c	c	c	c
202142	0-30, 75%, no replay	c	c	c	c	c	c	c	c	c
203121	0-40, 25%, replay	c	c	c	c	c	c	c	c	c
203122	0-40, 25%, no replay	c	c	c	c	c	c	c	c	c
203131	0-40, 25%, replay	c	c	c	c	c	c	c	c	c
203132	0-40, 25%, no replay	c	c	c	c	c	c	c	c	c
203141	0-40, 25%, replay	c	c	c	c	c	c	c	c	c
203142	0-40, 25%, no replay	c	c	c	c	c	c	c	c	c
210121	15-0, 25%, replay	nc	nc	nc	nc	nc	nc	nc	nc	nc
210122	15-0, 25%, no replay	c	c	nc	nc	nc	nc	nc	nc	nc
210131	15-0, 25%, replay	nc	nc	nc	nc	nc	nc	c	c	c
210132	15-0, 25%, replay	c	c	c	c	c	c	c	c	c
210141	15-0, 25%, no replay	c	c	c	c	c	c	c	c	c
210142	15-0, 25%, replay	c	c	c	c	c	c	c	c	c
211121	15-15, 25%, replay	nc	nc	nc	nc	nc	nc	nc	nc	nc
211122	15-15, 25%, no replay	c	c	c	nc	nc	nc	nc	nc	nc
211131	15-15, 50%, replay	c	c	c	c	c	c	c	c	c
211132	15-15, 50%, no replay	c	c	c	c	c	c	c	c	c
211141	15-15, 75%, replay	c	c	c	c	c	c	c	c	c
211142	15-15, 75%, no replay	c	c	c	c	c	c	c	c	c
212121	15-30, 25%, replay	c	c	nc	nc	nc	nc	nc	nc	nc
212122	15-30, 25%, no replay	c	c	c	c	c	c	c	c	c
212131	15-30, 50%, replay	c	c	c	c	c	c	c	c	c
212132	15-30, 50%, no replay	c	c	c	c	c	c	c	c	c
212141	15-30, 75%, replay	c	c	c	c	c	c	c	c	c
212142	15-30, 75%, no replay	c	c	c	c	c	c	c	c	c
213121	15-40, 25%, replay	c	c	c	c	c	c	c	c	c

Continued on Next Page. . .

Table 3.3 – Continued

		p								
States	**Score, Perception, Result of successful challenge**	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
213122	15-40, 25%, no replay	c	c	c	c	c	c	c	c	c
213131	15-40, 50%, replay	c	c	c	c	c	c	c	c	c
213132	15-40, 50%, no replay	c	c	c	c	c	c	c	c	c
213141	15-40, 75%, replay	c	c	c	c	c	c	c	c	c
213142	15-40, 75%, no replay	c	c	c	c	c	c	c	c	c
220121	30-0, 25%, replay	nc	nc	nc	nc	nc	nc	nc	nc	nc
220122	30-0, 25%, no replay	c	nc	nc	nc	nc	nc	nc	nc	nc
220131	30-0, 50%, replay	nc	nc	nc	nc	nc	nc	nc	c	c
220132	30-0, 50%, no replay	c	c	c	c	c	c	c	c	c
220141	30-0, 75%, replay	c	c	c	c	c	c	c	c	c
220142	30-0, 75%, no replay	c	c	c	c	c	c	c	c	c
221121	30-15, 25%, replay	nc	nc	nc	nc	nc	nc	nc	nc	nc
221122	30-15, 25%, no replay	c	c	c	nc	nc	nc	nc	nc	nc
221131	30-15, 50%, replay	c	c	c	c	c	c	c	c	c
221132	30-15, 50%, no replay	c	c	c	c	c	c	c	c	c
221141	30-15, 75%, replay	c	c	c	c	c	c	c	c	c
221142	30-15, 75%, no replay	c	c	c	c	c	c	c	c	c
222121	30-30, 25%, replay	c	nc	nc	nc	nc	nc	nc	nc	nc
222122	30-30, 25%, no replay	c	c	c	c	c	c	nc	nc	nc
222131	30-30, 50%, replay	c	c	c	c	c	c	c	c	c
222132	30-30, 50%, no replay	c	c	c	c	c	c	c	c	c
222141	30-30, 75%, replay	c	c	c	c	c	c	c	c	c
222142	30-30, 75%, no replay	c	c	c	c	c	c	c	c	c
223121	30-40, 25%, replay	c	c	c	c	c	c	c	c	c
223122	30-40, 25%, no replay	c	c	c	c	c	c	c	c	c
223131	30-40, 50%, replay	c	c	c	c	c	c	c	c	c
223132	30-40, 50%, no replay	c	c	c	c	c	c	c	c	c
223141	30-40, 75%, replay	c	c	c	c	c	c	c	c	c
223142	30-40, 75%, no replay	c	c	c	c	c	c	c	c	c

Continued on Next Page. . .

Table 3.3 – Continued

		p								
States	**Score, Perception, Result of successful challenge**	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
230121	40-0, 25%, replay	nc	nc	nc	nc	nc	nc	nc	nc	nc
230122	40-0, 25%, no replay	c	nc	nc	nc	nc	nc	nc	nc	nc
230131	40-0, 50%, replay	nc	nc	nc	nc	nc	nc	nc	nc	nc
230132	40-0, 50%, no replay	c	c	c	c	c	c	c	c	nc
230141	40-0, 75%, replay	c	c	c	c	c	c	c	c	c
230142	40-0, 75%, no replay	c	c	c	c	c	c	c	c	c
231121	40-15, 25%, replay	nc	nc	nc	nc	nc	nc	nc	nc	nc
231122	40-15, 25%, no replay	c	c	nc	nc	nc	nc	nc	nc	nc
231131	40-15, 50%, replay	nc	c	c	c	c	nc	c	c	c
231132	40-15, 50%, no replay	c	c	c	c	c	c	c	c	c
231141	40-15, 75%, replay	c	c	c	c	c	c	c	c	c
231142	40-15, 75%, no replay	c	c	c	c	c	c	c	c	c
232121	40-30, 25%, replay	c	nc	nc	nc	nc	nc	nc	nc	nc
232122	40-30, 25%, no replay	c	c	c	c	nc	nc	nc	nc	nc
232131	40-30, 50%, replay	c	c	c	c	c	c	c	c	c
232132	40-30, 50%, no replay	c	c	c	c	c	c	c	c	c
232141	40-30, 75%, replay	c	c	c	c	c	c	c	c	c
232142	40-30, 75%, no replay	c	c	c	c	c	c	c	c	c
233121	deuce, 25%, replay	c	nc	nc	nc	nc	nc	nc	nc	nc
233122	deuce, 25%, no replay	c	c	c	c	c	c	nc	nc	nc
233131	deuce, 50%, replay	c	c	c	c	c	c	c	c	c
233132	deuce, 50%, no replay	c	c	c	c	c	c	c	c	c
233141	deuce, 75%, replay	c	c	c	c	c	c	c	c	c
233142	deuce, 75%, no replay	c	c	c	c	c	c	c	c	c
234121	ad out, 25%, replay	c	c	c	c	c	c	c	c	c
234122	ad out, 25%, no replay	c	c	c	c	c	c	c	c	c
234131	ad out, 50%, replay	c	c	c	c	c	c	c	c	c
234132	ad out, 50%, no replay	c	c	c	c	c	c	c	c	c

Continued on Next Page. . .

Table 3.3 – Continued

		p								
States	**Score, Perception, Result of successful challenge**	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
234141	ad out, 75%, replay	c	c	c	c	c	c	c	c	c
234142	ad out, 75%, no replay	c	c	c	c	c	c	c	c	c
243121	ad in, 25%, replay	c	nc	nc	nc	nc	nc	nc	nc	nc
243122	ad in, 25%, no replay	c	c	c	c	nc	nc	nc	nc	nc
243131	ad in, 50%, replay	c	c	c	c	c	c	c	c	c
243132	ad in, 50%, no replay	c	c	c	c	c	c	c	c	c
243141	ad in, 75%, replay	c	c	c	c	c	c	c	c	c
243142	ad in, 75%, no replay	c	c	c	c	c	c	c	c	c

3.3.3 Case 3: Value of p_1 is 0.1

The results in this case are shown in Table 3.4 below. These can be interpreted in the same way as we did for the case when the value of p_1 is 0.5. The only difference is that in this case there are more states in which the player does not challenge because the proportion of time the player ends up in a state where his perception of winning is positive is higher as compared to the case when p_1 is 0.1.

Table 3.4: Effect of varying p when p_1 is 0.1

		p								
States	**Score, Perception, Result of successful challenge**	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
200121	0-0, 25%, replay	nc	nc	nc	nc	nc	nc	nc	nc	nc

Continued on Next Page...

Table 3.4 – Continued

		p								
States	**Score, Perception, Result of successful challenge**	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
200122	0-0, 25%, no replay	nc	nc	nc	nc	nc	nc	nc	nc	nc
200131	0-0, 50%, replay	nc	nc	nc	nc	nc	nc	nc	nc	nc
200132	0-0, 50%, no replay	c	c	c	c	c	c	nc	nc	nc
200141	0-0, 75%, replay	c	c	c	c	c	c	c	c	c
200142	0-0, 75%, no replay	c	c	c	c	c	c	c	c	c
201121	0-15, 25%, replay	nc	nc	nc	nc	nc	nc	nc	nc	nc
201122	0-15, 25%, no replay	c	nc	nc	nc	nc	nc	nc	nc	nc
201131	0-15, 50%, replay	nc	nc	nc	nc	nc	nc	nc	nc	nc
201132	0-15, 50%, no replay	c	c	c	c	c	c	c	nc	nc
201141	0-15, 75%, replay	c	c	c	c	c	c	c	c	c
201142	0-15, 75%, no replay	c	c	c	c	c	c	c	c	c
202121	0-30, 25%, replay	nc	nc	nc	nc	nc	nc	nc	nc	nc
202122	0-30, 25%, no replay	c	c	c	nc	nc	nc	nc	nc	nc
202131	0-30, 50%, replay	c	c	c	c	c	c	c	c	c
202132	0-30, 50%, no replay	c	c	c	c	c	c	c	c	c
202141	0-30, 75%, replay	c	c	c	c	c	c	c	c	c
202142	0-30, 75%, no replay	c	c	c	c	c	c	c	c	c
203121	0-40, 25%, replay	c	c	c	c	c	c	c	c	c
203122	0-40, 25%, no replay	c	c	c	c	c	c	c	c	c
203131	0-40, 50%, replay	c	c	c	c	c	c	c	c	c
203132	0-40, 50%, no replay	c	c	c	c	c	c	c	c	c
203141	0-40, 75%, replay	c	c	c	c	c	c	c	c	c
203142	0-40, 75%, no replay	c	c	c	c	c	c	c	c	c
210121	15-0, 25%, replay	nc	nc	nc	nc	nc	nc	nc	nc	nc
210122	15-0, 25%, no replay	nc	nc	nc	nc	nc	nc	nc	nc	nc
210131	15-0, 50%, replay	nc	nc	nc	nc	nc	nc	nc	nc	nc
210132	15-0, 50%, no replay	c	c	c	c	nc	nc	nc	nc	nc
210141	15-0, 75%, replay	c	c	c	c	c	c	c	c	c
210142	15-0, 75%, no replay	c	c	c	c	c	c	c	c	c

Continued on Next Page...

Table 3.4 – Continued

States	Score, Perception, Result of successful challenge	p 0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
211121	15-15, 25%, replay	nc	nc	nc	nc	nc	nc	nc	nc	nc
211122	15-15, 25%, no replay	c	nc	nc	nc	nc	nc	nc	nc	nc
211131	15-15, 50%, replay	nc	nc	nc	nc	nc	nc	nc	nc	nc
211132	15-15, 50%, no replay	c	c	c	c	c	c	nc	nc	nc
211141	15-15, 75%, replay	c	c	c	c	c	c	c	c	c
211142	15-15, 75%, no replay	c	c	c	c	c	c	c	c	c
212121	15-30, 25%, replay	nc	nc	nc	nc	nc	nc	nc	nc	nc
212122	15-30, 25%, no replay	c	c	c	nc	nc	nc	nc	nc	nc
212131	15-30, 50%, replay	c	c	c	c	c	c	c	c	c
212132	15-30, 50%, no replay	c	c	c	c	c	c	c	c	c
212141	15-30, 75%, replay	c	c	c	c	c	c	c	c	c
212142	15-30, 75%, no replay	c	c	c	c	c	c	c	c	c
213121	15-40, 25%, replay	c	c	c	c	c	c	c	c	c
213122	15-40, 25%, no replay	c	c	c	c	c	c	c	c	c
213131	15-40, 50%, replay	c	c	c	c	c	c	c	c	c
213132	15-40, 50%, no replay	c	c	c	c	c	c	c	c	c
213141	15-40, 75%, replay	c	c	c	c	c	c	c	c	c
213142	15-40, 75%, no replay	c	c	c	c	c	c	c	c	c
220121	30-0, 25%, replay	nc	nc	nc	nc	nc	nc	nc	nc	nc
220122	30-0, 25%, no replay	nc	nc	nc	nc	nc	nc	nc	nc	nc
220131	30-0, 50%, replay	nc	nc	nc	nc	nc	nc	nc	c	c
220132	30-0, 50%, no replay	c	c	c	c	nc	nc	nc	nc	nc
220141	30-0, 75%, replay	c	c	c	c	c	c	c	c	c
220142	30-0, 75%, no replay	c	c	c	c	c	c	c	c	c
221121	30-15, 25%, replay	nc	nc	nc	nc	nc	nc	nc	nc	nc
221122	30-15, 25%, no replay	nc	nc	nc	nc	nc	nc	nc	nc	nc
221131	30-15, 50%, replay	nc	nc	nc	nc	nc	nc	nc	nc	nc
221132	30-15, 50%, no replay	c	c	c	c	c	c	nc	nc	nc

Continued on Next Page...

Table 3.4 – Continued

		p								
States	**Score, Perception, Result of successful challenge**	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
221141	30-15, 75%, replay	c	c	c	c	c	c	c	c	c
221142	30-15, 75%, no replay	c	c	c	c	c	c	c	c	c
222121	30-30, 25%, replay	nc	nc	nc	nc	nc	nc	nc	nc	nc
222122	30-30, 25%, no replay	c	c	nc	nc	nc	nc	nc	nc	nc
222131	30-30, 50%, replay	c	c	c	c	nc	nc	nc	c	c
222132	30-30, 50%, no replay	c	c	c	c	c	c	c	c	c
222141	30-30, 75%, replay	c	c	c	c	c	c	c	c	c
222142	30-30, 75%, no replay	c	c	c	c	c	c	c	c	c
223121	30-40, 25%, replay	c	c	c	c	c	c	c	c	c
223122	30-40, 25%, no replay	c	c	c	c	c	c	c	c	c
223131	30-40, 50%, replay	c	c	c	c	c	c	c	c	c
223132	30-40, 50%, no replay	c	c	c	c	c	c	c	c	c
223141	30-40, 75%, replay	c	c	c	c	c	c	c	c	c
223142	30-40, 75%, no replay	c	c	c	c	c	c	c	c	c
230121	40-0, 25%, replay	nc	nc	nc	nc	nc	nc	nc	nc	nc
230122	40-0, 25%, no replay	nc	nc	nc	nc	nc	nc	nc	nc	nc
230131	40-0, 50%, replay	nc	nc	nc	nc	nc	nc	nc	nc	nc
230132	40-0, 50%, no replay	c	c	c	c	nc	nc	nc	nc	nc
230141	40-0, 75%, replay	nc	c	c	c	c	c	c	c	c
230142	40-0, 75%, no replay	c	c	c	c	c	c	c	c	c
231121	40-15, 25%, replay	nc	nc	nc	nc	nc	nc	nc	nc	nc
231122	40-15, 25%, no replay	nc	nc	nc	nc	nc	nc	nc	nc	nc
231131	40-15, 50%, replay	nc	nc	nc	nc	nc	nc	nc	nc	nc
231132	40-15, 50%, no replay	c	c	c	c	c	nc	nc	nc	nc
231141	40-15, 75%, replay	c	c	c	c	c	c	c	c	c
231142	40-15, 75%, no replay	c	c	c	c	c	c	c	c	c
232121	40-30, 25%, replay	nc	nc	nc	nc	nc	nc	nc	nc	nc
232122	40-30, 25%, no replay	c	c	nc	nc	nc	nc	nc	nc	nc

Continued on Next Page...

Table 3.4 – Continued

		p								
States	**Score, Perception, Result of successful challenge**	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
232131	40-30, 50%, replay	c	c	c	nc	nc	nc	nc	nc	nc
232132	40-30, 50%, no replay	c	c	c	c	c	c	nc	nc	nc
232141	40-30, 75%, replay	c	c	c	c	c	c	c	c	c
232142	40-30, 75%, no replay	c	c	c	c	c	c	c	c	c
233121	deuce, 25%, replay	nc	nc	nc	nc	nc	nc	nc	nc	nc
233122	deuce, 25%, no replay	c	c	nc	nc	nc	nc	nc	nc	nc
233131	deuce, 50%, replay	c	c	c	c	nc	nc	nc	c	c
233132	deuce, 50%, no replay	c	c	c	c	c	c	c	c	c
233141	deuce, 75%, replay	c	c	c	c	c	c	c	c	c
233142	deuce, 75%, no replay	c	c	c	c	c	c	c	c	c
234121	ad out, 25%, replay	c	c	c	c	c	c	c	c	c
234122	ad out, 25%, no replay	c	c	c	c	c	c	c	c	c
234131	ad out, 50%, replay	c	c	c	c	c	c	c	c	c
234132	ad out, 50%, no replay	c	c	c	c	c	c	c	c	c
234141	ad out, 75%, replay	c	c	c	c	c	c	c	c	c
234142	ad out, 75%, no replay	c	c	c	c	c	c	c	c	c
243121	ad in, 25%, replay	nc	nc	nc	nc	nc	nc	nc	nc	nc
243122	ad in, 25%, no replay	c	c	nc	nc	nc	nc	nc	nc	nc
243131	ad in, 50%, replay	c	c	c	nc	nc	nc	nc	nc	nc
243132	ad in, 50%, no replay	c	c	c	c	c	c	nc	nc	nc
243141	ad in, 75%, replay	c	c	c	c	c	c	c	c	c
243142	ad in, 75%, no replay	c	c	c	c	c	c	c	c	c

3.4 Effect of varying s

In this experiment the proportion of time that a player ends up in a state where his perception of winning a challenge is 0%, i.e., p_1 is fixed at 0.9.

The remaining time is divided equally between states where perception of winning a challenge is s_2 or s_3 or s_4 percent, i.e, $p_2 = p_3 = p_4 = (1 - p_1)/3$. Each player has four levels of perception s_1, s_2, s_3 and s_4. Our motive is to study the effect of varying these preception level parameters. To study this three different sub-experiments were performed where in each sub-experiment the perception levels are fixed. The results of these sub-experiments are presented in Tables 3.2, 3.5 and 3.6. The proportion of time the point will be replayed in the case of a correct challenge r is fixed at 0.6 and player's probability of winning a point is varied from 0.1 to 0.9. The value of p_1 is varied across the column and each row corresponds to a different state. The values in the table show the optimal action for each combination of state and p.

Table 3.5: Perceptions levels 0%, 5%, 15% and 25%

		p_1								
States	**Score, Perception, Result of successful challenge**	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
200121	0-0, 5%, replay	nc	nc	nc	nc	nc	nc	nc	c	c
200122	0-0, 5%, no replay	c	c	c	c	c	c	c	c	c
200131	0-0, 15%, replay	c	c	c	c	c	c	c	c	c
200132	0-0, 15%, no replay	c	c	c	c	c	c	c	c	c
200141	0-0, 25%, replay	c	c	c	c	c	c	c	c	c
200142	0-0, 25%, no replay	c	c	c	c	c	c	c	c	c
201121	0-15, 5%, replay	nc	nc	nc	c	c	c	c	c	c
201122	0-15, 5%, no replay	c	c	c	c	c	c	c	c	c
201131	0-15, 15%, replay	c	c	c	c	c	c	c	c	c
201132	0-15, 15%, no replay	c	c	c	c	c	c	c	c	c
201141	0-15, 25%, replay	c	c	c	c	c	c	c	c	c
201142	0-15, 25%, no replay	c	c	c	c	c	c	c	c	c

Continued on Next Page...

Table 3.5 – Continued

States	Score, Perception, Result of successful challenge	p_1 0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
202121	0-30, 5%, replay	c	c	c	c	c	c	c	c	c
202122	0-30, 5%, no replay	c	c	c	c	c	c	c	c	c
202131	0-30, 15%, replay	c	c	c	c	c	c	c	c	c
202132	0-30, 15%, no replay	c	c	c	c	c	c	c	c	c
202141	0-30, 25%, replay	c	c	c	c	c	c	c	c	c
202142	0-30, 25%, no replay	c	c	c	c	c	c	c	c	c
203121	0-40, 5%, replay	c	c	c	c	c	c	c	c	c
203122	0-40, 5%, no replay	c	c	c	c	c	c	c	c	c
203131	0-40, 15%, replay	c	c	c	c	c	c	c	c	c
203132	0-40, 15%, no replay	c	c	c	c	c	c	c	c	c
203141	0-40, 25%, replay	c	c	c	c	c	c	c	c	c
203142	0-40, 25%, no replay	c	c	c	c	c	c	c	c	c
210121	15-0, 5%, replay	nc	nc	nc	nc	nc	nc	nc	c	c
210122	15-0, 5%, no replay	c	c	c	c	c	c	c	c	c
210131	15-0, 15%, replay	c	c	c	c	c	c	c	c	c
210132	15-0, 15%, no replay	c	c	c	c	c	c	c	c	c
210141	15-0, 25%, replay	c	c	c	c	c	c	c	c	c
210142	15-0, 25%, no replay	c	c	c	c	c	c	c	c	c
211121	15-15, 5%, replay	nc	nc	nc	c	c	c	c	c	c
211122	15-15, 5%, no replay	c	c	c	c	c	c	c	c	c
211131	15-15, 15%, replay	c	c	c	c	c	c	c	c	c
211132	15-15, 15%, no replay	c	c	c	c	c	c	c	c	c
211141	15-15, 25%, replay	c	c	c	c	c	c	c	c	c
211142	15-15, 25%, no replay	c	c	c	c	c	c	c	c	c
212121	15-30, 5%, replay	c	c	c	c	c	c	c	c	c
212122	15-30, 5%, no replay	c	c	c	c	c	c	c	c	c
212131	15-30, 15%, replay	c	c	c	c	c	c	c	c	c
212132	15-30, 15%, no replay	c	c	c	c	c	c	c	c	c
212141	15-30, 25%, replay	c	c	c	c	c	c	c	c	c

Continued on Next Page...

Table 3.5 – Continued

		p_1								
States	**Score, Perception, Result of successful challenge**	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
212142	15-30, 25%, no replay	c	c	c	c	c	c	c	c	c
213121	15-40, 5%, replay	c	c	c	c	c	c	c	c	c
213122	15-40, 5%, no replay	c	c	c	c	c	c	c	c	c
213131	15-40, 15%, replay	c	c	c	c	c	c	c	c	c
213132	15-40, 15%, no replay	c	c	c	c	c	c	c	c	c
213141	15-40, 25%, replay	c	c	c	c	c	c	c	c	c
213142	15-40, 25%, no replay	c	c	c	c	c	c	c	c	c
220121	30-0, 5%, replay	nc	nc	nc	nc	nc	nc	nc	nc	c
220122	30-0, 5%, no replay	c	c	c	c	c	c	c	c	c
220131	30-0, 15%, replay	c	c	c	c	c	c	c	c	c
220132	30-0, 15%, no replay	c	c	c	c	c	c	c	c	c
220141	30-0, 25%, replay	c	c	c	c	c	c	c	c	c
220142	30-0, 25%, no replay	c	c	c	c	c	c	c	c	c
221121	30-15, 5%, replay	nc	nc	nc	nc	c	c	c	c	c
221122	30-15, 5%, no replay	c	c	c	c	c	c	c	c	c
221131	30-15, 15%, replay	c	c	c	c	c	c	c	c	c
221132	30-15, 15%, no replay	c	c	c	c	c	c	c	c	c
221141	30-15, 25%, replay	c	c	c	c	c	c	c	c	c
221142	30-15, 25%, no replay	c	c	c	c	c	c	c	c	c
222121	30-30, 5%, replay	c	c	c	c	c	c	c	c	c
222122	30-30, 5%, no replay	c	c	c	c	c	c	c	c	c
222131	30-30, 15%, replay	c	c	c	c	c	c	c	c	c
222132	30-30, 15%, no replay	c	c	c	c	c	c	c	c	c
222141	30-30, 25%, replay	c	c	c	c	c	c	c	c	c
222142	30-30, 25%, no replay	c	c	c	c	c	c	c	c	c
223121	30-40, 5%, replay	c	c	c	c	c	c	c	c	c
223122	30-40, 5%, no replay	c	c	c	c	c	c	c	c	c
223131	30-40, 15%, replay	c	c	c	c	c	c	c	c	c

Continued on Next Page. . .

Table 3.5 – Continued

States	Score, Perception, Result of successful challenge	p_1 0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
223132	30-40, 15%, no replay	c	c	c	c	c	c	c	c	c
223141	30-40, 25%, replay	c	c	c	c	c	c	c	c	c
223142	30-40, 25%, no replay	c	c	c	c	c	c	c	c	c
230121	40-0, 5%, replay	nc	nc	nc	nc	nc	nc	nc	nc	nc
230122	40-0, 5%, no replay	c	c	c	c	c	c	c	c	nc
230131	40-0, 15%, replay	c	c	c	c	c	c	c	c	c
230132	40-0, 15%, no replay	c	c	c	c	c	c	c	c	c
230141	40-0, 25%, replay	c	c	c	c	c	c	c	c	c
230142	40-0, 25%, no replay	c	c	c	c	c	c	c	c	c
231121	40-15, 5%, replay	nc	nc	nc	nc	nc	nc	nc	c	c
231122	40-15, 5%, no replay	c	c	c	c	c	c	c	c	c
231131	40-15, 15%, replay	c	c	c	c	c	c	c	c	c
231132	40-15, 15%, no replay	c	c	c	c	c	c	c	c	c
231141	40-15, 25%, replay	c	c	c	c	c	c	c	c	c
231142	40-15, 25%, no replay	c	c	c	c	c	c	c	c	c
232121	40-30, 5%, replay	c	c	c	c	c	c	c	c	c
232122	40-30, 5%, no replay	c	c	c	c	c	c	c	c	c
232131	40-30, 15%, replay	c	c	c	c	c	c	c	c	c
232132	40-30, 15%, no replay	c	c	c	c	c	c	c	c	c
232141	40-30, 25%, replay	c	c	c	c	c	c	c	c	c
232142	40-30, 25%, no replay	c	c	c	c	c	c	c	c	c
233121	deuce, 5%, replay	c	c	c	c	c	c	c	c	c
233122	deuce, 5%, no replay	c	c	c	c	c	c	c	c	c
233131	deuce, 15%, replay	c	c	c	c	c	c	c	c	c
233132	deuce, 15%, no replay	c	c	c	c	c	c	c	c	c
233141	deuce, 25%, replay	c	c	c	c	c	c	c	c	c
233142	deuce, 25%, no replay	c	c	c	c	c	c	c	c	c
234121	ad out, 5%, replay	c	c	c	c	c	c	c	c	c

Continued on Next Page. . .

Table 3.5 – Continued

		p_1								
States	**Score, Perception, Result of successful challenge**	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
234122	ad out, 5%, no replay	c	c	c	c	c	c	c	c	c
234131	ad out, 15%, replay	c	c	c	c	c	c	c	c	c
234132	ad out, 15%, no replay	c	c	c	c	c	c	c	c	c
234141	ad out, 25%, replay	c	c	c	c	c	c	c	c	c
234142	ad out, 25%, no replay	c	c	c	c	c	c	c	c	c
243121	ad in, 5%, replay	c	c	c	c	c	c	c	c	c
243122	ad in, 5%, no replay	c	c	c	c	c	c	c	c	c
243131	ad in, 15%, replay	c	c	c	c	c	c	c	c	c
243132	ad in, 15%, no replay	c	c	c	c	c	c	c	c	c
243141	ad in, 25%, replay	c	c	c	c	c	c	c	c	c
243142	ad in, 25%, no replay	c	c	c	c	c	c	c	c	c

Table 3.6: Perceptions levels 0%, 5%, 10% and 15%

		p_1								
States	**Score, Perception, Result of succesful challenge**	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
200121	0-0, 5%, replay	nc	nc	nc	nc	nc	nc	nc	c	c
200122	0-0, 5%, no replay	c	c	c	c	c	c	c	c	c
200131	0-0, 10%, replay	c	c	c	c	c	c	c	c	c
200132	0-0, 10%, no replay	c	c	c	c	c	c	c	c	c
200141	0-0, 15%, replay	c	c	c	c	c	c	c	c	c
200142	0-0, 15%, no replay	c	c	c	c	c	c	c	c	c
201121	0-15, 5%, replay	nc	nc	nc	c	c	c	c	c	c
201122	0-15, 5%, no replay	c	c	c	c	c	c	c	c	c
201131	0-15, 10%, replay	c	c	c	c	c	c	c	c	c
201132	0-15, 10%, no replay	c	c	c	c	c	c	c	c	c

Continued on Next Page...

Table 3.6 – Continued

States	Score, Perception, Result of succesful challenge	p_1 0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
201141	0-15, 15%, replay	c	c	c	c	c	c	c	c	c
201142	0-15, 15%, no replay	c	c	c	c	c	c	c	c	c
202121	0-30, 5%, replay	c	c	c	c	c	c	c	c	c
202122	0-30, 5%, no replay	c	c	c	c	c	c	c	c	c
202131	0-30, 10%, replay	c	c	c	c	c	c	c	c	c
202132	0-30, 10%, no replay	c	c	c	c	c	c	c	c	c
202141	0-30, 15%, replay	c	c	c	c	c	c	c	c	c
202142	0-30, 15%, no replay	c	c	c	c	c	c	c	c	c
203121	0-40, 5%, replay	c	c	c	c	c	c	c	c	c
203122	0-40, 5%, no replay	c	c	c	c	c	c	c	c	c
203131	0-40, 10%, replay	c	c	c	c	c	c	c	c	c
203132	0-40, 10%, no replay	c	c	c	c	c	c	c	c	c
203141	0-40, 15%, replay	c	c	c	c	c	c	c	c	c
203142	0-40, 15%, no replay	c	c	c	c	c	c	c	c	c
210121	15-0, 5%, replay	nc	nc	nc	nc	nc	nc	nc	c	c
210122	15-0, 5%, no replay	c	c	c	c	c	c	c	c	c
210131	15-0, 10%, replay	c	c	c	c	c	c	c	c	c
210132	15-0, 10%, no replay	c	c	c	c	c	c	c	c	c
210141	15-0, 15%, replay	c	c	c	c	c	c	c	c	c
210142	15-0, 15%, no replay	c	c	c	c	c	c	c	c	c
211121	15-15, 5%, replay	nc	nc	nc	c	c	c	c	c	c
211122	15-15, 5%, no replay	c	c	c	c	c	c	c	c	c
211131	15-15, 10%, replay	c	c	c	c	c	c	c	c	c
211132	15-15, 10%, no replay	c	c	c	c	c	c	c	c	c
211141	15-15, 15%, replay	c	c	c	c	c	c	c	c	c
211142	15-15, 15%, no replay	c	c	c	c	c	c	c	c	c
212121	15-30, 5%, replay	c	c	c	c	c	c	c	c	c
212122	15-30, 5%, no replay	c	c	c	c	c	c	c	c	c

Continued on Next Page...

Table 3.6 – Continued

		p_1								
States	**Score, Perception, Result of succesful challenge**	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
212131	15-30, 10%, replay	c	c	c	c	c	c	c	c	c
212132	15-30, 10%, no replay	c	c	c	c	c	c	c	c	c
212141	15-30, 15%, replay	c	c	c	c	c	c	c	c	c
212142	15-30, 15%, no replay	c	c	c	c	c	c	c	c	c
213121	15-40, 5%, replay	c	c	c	c	c	c	c	c	c
213122	15-40, 5%, no replay	c	c	c	c	c	c	c	c	c
213131	15-40, 10%, replay	c	c	c	c	c	c	c	c	c
213132	15-40, 10%, no replay	c	c	c	c	c	c	c	c	c
213141	15-40, 15%, replay	c	c	c	c	c	c	c	c	c
213142	15-40, 15%, no replay	c	c	c	c	c	c	c	c	c
220121	30-0, 5%, replay	nc	nc	nc	nc	nc	nc	nc	nc	c
220122	30-0, 5%, no replay	c	c	c	c	c	c	c	c	c
220131	30-0, 10%, replay	c	c	c	c	c	c	c	c	c
220132	30-0, 10%, no replay	c	c	c	c	c	c	c	c	c
220141	30-0, 15%, replay	c	c	c	c	c	c	c	c	c
220142	30-0, 15%, no replay	c	c	c	c	c	c	c	c	c
221121	30-15, 5%, replay	nc	nc	nc	nc	c	c	c	c	c
221122	30-15, 5%, no replay	c	c	c	c	c	c	c	c	c
221131	30-15, 10%, replay	c	c	c	c	c	c	c	c	c
221132	30-15, 10%, no replay	c	c	c	c	c	c	c	c	c
221141	30-15, 15%, replay	c	c	c	c	c	c	c	c	c
221142	30-15, 15%, no replay	c	c	c	c	c	c	c	c	c
222121	30-30, 5%, replay	c	c	c	c	c	c	c	c	c
222122	30-30, 5%, no replay	c	c	c	c	c	c	c	c	c
222131	30-30, 10%, replay	c	c	c	c	c	c	c	c	c
222132	30-30, 10%, no replay	c	c	c	c	c	c	c	c	c
222141	30-30, 15%, replay	c	c	c	c	c	c	c	c	c
222142	30-30, 15%, no replay	c	c	c	c	c	c	c	c	c

Continued on Next Page...

Table 3.6 – Continued

		p_1								
States	**Score, Perception, Result of succesful challenge**	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
223121	30-40, 5%, replay	c	c	c	c	c	c	c	c	c
223122	30-40, 5%, no replay	c	c	c	c	c	c	c	c	c
223131	30-40, 10%, replay	c	c	c	c	c	c	c	c	c
223132	30-40, 10%, no replay	c	c	c	c	c	c	c	c	c
223141	30-40, 15%, replay	c	c	c	c	c	c	c	c	c
223142	30-40, 15%, no replay	c	c	c	c	c	c	c	c	c
230121	40-0, 5%, replay	nc	nc	nc	nc	nc	nc	nc	nc	nc
230122	40-0, 5%, no replay	c	c	c	c	c	c	c	c	c
230131	40-0, 10%, replay	c	c	c	c	c	c	c	c	c
230132	40-0, 10%, no replay	c	c	c	c	c	c	c	c	c
230141	40-0, 15%, replay	c	c	c	c	c	c	c	c	c
230142	40-0, 15%, no replay	c	c	c	c	c	c	c	c	c
231121	40-15, 5%, replay	nc	nc	nc	nc	nc	nc	nc	c	c
231122	40-15, 5%, no replay	c	c	c	c	c	c	c	c	c
231131	40-15, 10%, replay	c	c	c	c	c	c	c	c	c
231132	40-15, 10%, no replay	c	c	c	c	c	c	c	c	c
231141	40-15, 15%, replay	c	c	c	c	c	c	c	c	c
231142	40-15, 15%, no replay	c	c	c	c	c	c	c	c	c
232121	40-30, 5%, replay	c	c	c	c	c	c	c	c	c
232122	40-30, 5%, no replay	c	c	c	c	c	c	c	c	c
232131	40-30, 10%, replay	c	c	c	c	c	c	c	c	c
232132	40-30, 10%, no replay	c	c	c	c	c	c	c	c	c
232141	40-30, 15%, replay	c	c	c	c	c	c	c	c	c
232142	40-30, 15%, no replay	c	c	c	c	c	c	c	c	c
233121	deuce, 5%, replay	c	c	c	c	c	c	c	c	c
233122	deuce, 5%, no replay	c	c	c	c	c	c	c	c	c
233131	deuce, 10%, replay	c	c	c	c	c	c	c	c	c
233132	deuce, 10%, no replay	c	c	c	c	c	c	c	c	c
233141	deuce, 15%, replay	c	c	c	c	c	c	c	c	c

Continued on Next Page...

Table 3.6 – Continued

		p_1								
States	**Score, Perception, Result of succesful challenge**	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
233142	deuce, 15%, no replay	c	c	c	c	c	c	c	c	c
234121	ad out, 5%, replay	c	c	c	c	c	c	c	c	c
234122	ad out, 5%, no replay	c	c	c	c	c	c	c	c	c
234131	ad out, 10%, replay	c	c	c	c	c	c	c	c	c
234132	ad out, 10%, no replay	c	c	c	c	c	c	c	c	c
234141	ad out, 15%, replay	c	c	c	c	c	c	c	c	c
234142	ad out, 15%, no replay	c	c	c	c	c	c	c	c	c
243121	ad in, 5%, replay	c	c	c	c	c	c	c	c	c
243122	ad in, 5%, no replay	c	c	c	c	c	c	c	c	c
243131	ad in, 10%, replay	c	c	c	c	c	c	c	c	c
243132	ad in, 10%, no replay	c	c	c	c	c	c	c	c	c
243141	ad in, 15%, replay	c	c	c	c	c	c	c	c	c
243142	ad in, 15%, no replay	c	c	c	c	c	c	c	c	c

From the results in the Tables 3.2, 3.5 and 3.6, we can see that the optimal action when the value of p_1 is 0.9 is to always challenge except in some states (rows corresponding to such states were highlighted in green in the Tables 3.2, 3.5 and 3.6) where the perception is as low as 5%. For example, the state 221121 in Table 3.6 is the one where the player lost the current play when the score is 30-15, the perception of winning a challenge is 5% and the point will be replayed if there is a correct challenge.

Chapter 4

Conclusions and future work

The model proposed in this thesis gives a good insight of when to use the limited opportunities to challenge an umpire's call in a normal game of a tennis. We observed that as the player's probability of seeing a challengable point is low he challenges and when it is above a threshold value he does not challenge. For a particular score we see that the higher the perception of winning, the better it is to challenge.

We have also observed that as a player becomes stronger (relative to his opponent), it becomes unnecessary for him to use his challenges. There are some exceptional states where the player has to challenge if he is either too weak or too strong. This makes sense because if he is too weak, challenging is the only way he could win a point and if he is too strong, losing challenges does not make as much difference to him.

We saw that when the proportion of time the player ends up in a state where he has a positive perception of winning a challenge is very low, he will challenge whenever he has a chance except in a very few situations such as when his perception of winning a challenge is very low.

A meaningful modification to this model is to extend it to include

a whole set. Our computational results examine the case where the player has just one challenge left per game. Obviously, it would be interesting to do further experiments to examine what happens when more challenges are available. Extending our model to a whole set involves modeling of a tie-breaker game which has complex serve change rules. This would require more modeling and computational effort. As our model considers only a normal game where a single player serves all the time, such an extension would be interesting. The player's probability of winning a point is considered to be homogeneous throughout a game in our model, which could be modified to come close to reality where probabilities vary from point to point. The player's perception in our model is split into four different levels and only a few values were considered in our experiments. A possible extension would be considering more discrete perception levels or making this variable to continuous because ultimately we are interested in looking at a mapping between the perception of the player and the optimal action at a more granular level.

Appendices

Appendix A

C++ code for genarating transition probability matrices

```
#include <iostream>
#include <fstream>
#include <string>
#include <vector>

using namespace std;


float p  = 0.6;
float p1 = 0.01;
float p2 = 0.33;
float p3 = 0.33;
float p4 = 0.33;
float s1 = 0.00;
float s2 = 0.25;
float s3 = 0.50;
float s4 = 0.75;
float k  = 0.6;


//function to extract the digit in ones place

int ones(int a)
{
        return a%10 ;
}


//function to extract the digit in tens place

int tens(int a)
{
        return (a/10)%10 ;
}


//function to extract the digit in hundreds place
int hundreds(int a)
{
        return (a/100)%10 ;
}


//function to remove the last 3 digits
int trunclast3(int a)
{
        return (a/1000);
}




//***function for generating trans prob matrix when you challenge**********

float  transprob_c(int a, int b)
```

```
{
        int d =0, e =0,f=0 ;
        float prob = 0.0 ;

        if((tens(trunclast3(a))== tens(trunclast3(b))) &&
        ((ones(trunclast3(b))- ones(trunclast3(a))) ==1))
                d = 1;
        if((tens(trunclast3(a))== tens(trunclast3(b))) &&
        ((ones(trunclast3(a))- ones(trunclast3(b))) ==1)  &&
         (tens(trunclast3(a))==3) && (ones(trunclast3(a))==4))
                e = 1;
        if((ones(trunclast3(a))== ones(trunclast3(b))) &&
         ((tens(trunclast3(a))- tens(trunclast3(b))) ==1)  &&
         (ones(trunclast3(a))==3) && (tens(trunclast3(a))==4))
                d = 1;
        if((ones(trunclast3(a))== ones(trunclast3(b))) &&
        ((tens(trunclast3(b))- tens(trunclast3(a))) ==1))
                e = 1;
        if ((tens(trunclast3(b))==4) && (tens(trunclast3(a))>= 3) &&
        (ones(trunclast3(a))!=4)&& (ones(trunclast3(b))!=3) &&
        (ones(trunclast3(a))!=tens(trunclast3(a))) )
                e = 1;
        if ((ones(trunclast3(b))==4) && (ones(trunclast3(a))>= 3) &&
        (tens(trunclast3(a))!=4) && (tens(trunclast3(b))!=3)&&
        (ones(trunclast3(a))!=tens(trunclast3(a))) )
                d = 1;
        if (  (tens(trunclast3(a))== tens(trunclast3(b))) &&
         ((ones(trunclast3(b))== ones( trunclast3(a) )  ) )   )
                f = 1;


        //when you win the challenge and there is a no replay

        if( e==1 && (hundreds(trunclast3(b)) == 1)  &&
         (hundreds(b)== 1 )&& (ones(a)== 2))
        {
                if(tens(a) == 1)
                        prob = p*s1 ;
                if(tens(a) == 2)
                        prob = p*s2 ;
                if(tens(a) == 3)
                        prob = p*s3 ;
                if(tens(a) == 4)
                        prob = p*s4 ;
        }

        if( e==1 && (hundreds(trunclast3(b)) == 2) && (ones(a)== 2))
        {
                if(ones(b)==1)
                {
                        if(tens(a) == 1)
                        {   if(tens(b) == 1)
                                prob = (1-p)*s1*k*p1 ;
                                if(tens(b) == 2)
                                prob = (1-p)*s1*k*p2 ;
                                if(tens(b) == 3)
                                prob = (1-p)*s1*k*p3 ;
                                if(tens(b) == 4)
                                prob = (1-p)*s1*k*p4 ;

                        }
                        if(tens(a) == 2)
                        {
                                if(tens(b) == 1)
                                prob = (1-p)*s2*k*p1 ;
                                if(tens(b) == 2)
                                prob = (1-p)*s2*k*p2 ;
                                if(tens(b) == 3)
                                prob = (1-p)*s2*k*p3 ;
                                if(tens(b) == 4)
                                prob = (1-p)*s2*k*p4 ;

                        }
                        if(tens(a) == 3)
                        {
                                if(tens(b) == 1)
```

```
                        prob = (1-p)*s3*k*p1 ;
                        if(tens(b) == 2)
                        prob = (1-p)*s3*k*p2 ;
                        if(tens(b) == 3)
                        prob = (1-p)*s3*k*p3 ;
                        if(tens(b) == 4)
                        prob = (1-p)*s3*k*p4 ;

                }
                if(tens(a) == 4)
                {
                        if(tens(b) == 1)
                        prob = (1-p)*s4*k*p1 ;
                        if(tens(b) == 2)
                        prob = (1-p)*s4*k*p2 ;
                        if(tens(b) == 3)
                        prob = (1-p)*s4*k*p3 ;
                        if(tens(b) == 4)
                        prob = (1-p)*s4*k*p4 ;

                }
        }

        if(ones(b)==2)
        {

                if(tens(a) == 1)
                {    if(tens(b) == 1)
                        prob = (1-p)*s1*(1-k)*p1 ;
                        if(tens(b) == 2)
                        prob = (1-p)*s1*(1-k)*p2 ;
                        if(tens(b) == 3)
                        prob = (1-p)*s1*(1-k)*p3 ;
                        if(tens(b) == 4)
                        prob = (1-p)*s1*(1-k)*p4 ;

                }
                if(tens(a) == 2)
                {
                        if(tens(b) == 1)
                        prob = (1-p)*s2*(1-k)*p1 ;
                        if(tens(b) == 2)
                        prob = (1-p)*s2*(1-k)*p2 ;
                        if(tens(b) == 3)
                        prob = (1-p)*s2*(1-k)*p3 ;
                        if(tens(b) == 4)
                        prob = (1-p)*s2*(1-k)*p4 ;

                }
                if(tens(a) == 3)
                {
                        if(tens(b) == 1)
                        prob = (1-p)*s3*(1-k)*p1 ;
                        if(tens(b) == 2)
                        prob = (1-p)*s3*(1-k)*p2 ;
                        if(tens(b) == 3)
                        prob = (1-p)*s3*(1-k)*p3 ;
                        if(tens(b) == 4)
                        prob = (1-p)*s3*(1-k)*p4 ;

                }
                if(tens(a) == 4)
                {
                        if(tens(b) == 1)
                        prob = (1-p)*s4*(1-k)*p1 ;
                        if(tens(b) == 2)
                        prob = (1-p)*s4*(1-k)*p2 ;
                        if(tens(b) == 3)
                        prob = (1-p)*s4*(1-k)*p3 ;
                        if(tens(b) == 4)
                        prob = (1-p)*s4*(1-k)*p4 ;

                }
        }
}
```

```
if( e==1 && (hundreds(trunclast3(b)) == 3)  &&
  (hundreds(b)== 1 ) && (ones(a)== 2) )
{
        if(tens(a) == 1)
                prob = s1 ;
        if(tens(a) == 2)
                prob = s2 ;
        if(tens(a) == 3)
                prob = s3 ;
        if(tens(a) == 4)
                prob = s4 ;
}



//when you win the challenge and there is a replay


if( f==1 && (hundreds(trunclast3(b)) == 1)  &&
(hundreds(b)== 1 )&& (ones(a)== 1))
{
        if(tens(a) == 1)
                prob = p*s1 ;
        if(tens(a) == 2)
                prob = p*s2 ;
        if(tens(a) == 3)
                prob = p*s3 ;
        if(tens(a) == 4)
                prob = p*s4 ;
}

if( f==1 && (hundreds(trunclast3(b)) == 2)
&& (ones(a)== 1))
{
        if(ones(b)==1)
        {
                if(tens(a) == 1)
                {   if(tens(b) == 1)
                        prob = (1-p)*s1*k*p1 ;
                        if(tens(b) == 2)
                        prob = (1-p)*s1*k*p2 ;
                        if(tens(b) == 3)
                        prob = (1-p)*s1*k*p3 ;
                        if(tens(b) == 4)
                        prob = (1-p)*s1*k*p4 ;

                }
                if(tens(a) == 2)
                {
                        if(tens(b) == 1)
                        prob = (1-p)*s2*k*p1 ;
                        if(tens(b) == 2)
                        prob = (1-p)*s2*k*p2 ;
                        if(tens(b) == 3)
                        prob = (1-p)*s2*k*p3 ;
                        if(tens(b) == 4)
                        prob = (1-p)*s2*k*p4 ;

                }
                if(tens(a) == 3)
                {
                        if(tens(b) == 1)
                        prob = (1-p)*s3*k*p1 ;
                        if(tens(b) == 2)
                        prob = (1-p)*s3*k*p2 ;
                        if(tens(b) == 3)
                        prob = (1-p)*s3*k*p3 ;
                        if(tens(b) == 4)
                        prob = (1-p)*s3*k*p4 ;

                }
                if(tens(a) == 4)
                {
                        if(tens(b) == 1)
```

```
                        prob = (1-p)*s4*k*p1 ;
                        if(tens(b) == 2)
                        prob = (1-p)*s4*k*p2 ;
                        if(tens(b) == 3)
                        prob = (1-p)*s4*k*p3 ;
                        if(tens(b) == 4)
                        prob = (1-p)*s4*k*p4 ;

                }
        }

        if(ones(b)==2)
        {

                if(tens(a) == 1)
                {    if(tens(b) == 1)
                        prob = (1-p)*s1*(1-k)*p1 ;
                        if(tens(b) == 2)
                        prob = (1-p)*s1*(1-k)*p2 ;
                        if(tens(b) == 3)
                        prob = (1-p)*s1*(1-k)*p3 ;
                        if(tens(b) == 4)
                        prob = (1-p)*s1*(1-k)*p4 ;

                }
                if(tens(a) == 2)
                {
                        if(tens(b) == 1)
                        prob = (1-p)*s2*(1-k)*p1 ;
                        if(tens(b) == 2)
                        prob = (1-p)*s2*(1-k)*p2 ;
                        if(tens(b) == 3)
                        prob = (1-p)*s2*(1-k)*p3 ;
                        if(tens(b) == 4)
                        prob = (1-p)*s2*(1-k)*p4 ;

                }
                if(tens(a) == 3)
                {
                        if(tens(b) == 1)
                        prob = (1-p)*s3*(1-k)*p1 ;
                        if(tens(b) == 2)
                        prob = (1-p)*s3*(1-k)*p2 ;
                        if(tens(b) == 3)
                        prob = (1-p)*s3*(1-k)*p3 ;
                        if(tens(b) == 4)
                        prob = (1-p)*s3*(1-k)*p4 ;

                }
                if(tens(a) == 4)
                {
                        if(tens(b) == 1)
                        prob = (1-p)*s4*(1-k)*p1 ;
                        if(tens(b) == 2)
                        prob = (1-p)*s4*(1-k)*p2 ;
                        if(tens(b) == 3)
                        prob = (1-p)*s4*(1-k)*p3 ;
                        if(tens(b) == 4)
                        prob = (1-p)*s4*(1-k)*p4 ;

                }
        }
}



//when you loose the challenge


if( d==1 && (hundreds(trunclast3(b)) == 1)  &&  (hundreds(b)== 0 ) )
{
        if(tens(a) == 1)
                prob = p*(1-s1) ;
        if(tens(a) == 2)
```

```
                prob = p*(1-s2) ;
        if(tens(a) == 3)
                prob = p*(1-s3) ;
        if(tens(a) == 4)
                prob = p*(1-s4) ;
}


if( d==1 && (hundreds(trunclast3(b)) == 2)  &&  (hundreds(b)== 0 ) )
{
        if(tens(a) == 1)
                prob = (1-p)*(1-s1) ;
        if(tens(a) == 2)
                prob = (1-p)*(1-s2) ;
        if(tens(a) == 3)
                prob = (1-p)*(1-s3) ;
        if(tens(a) == 4)
                prob = (1-p)*(1-s4) ;
}


if( d==1 && (hundreds(trunclast3(b)) == 3)  &&  (hundreds(b)== 0 ) )
{
        if(tens(a) == 1)
                prob =  (1-s1) ;
        if(tens(a) == 2)
                prob =  (1-s2) ;
        if(tens(a) == 3)
                prob =  (1-s3) ;
        if(tens(a) == 4)
                prob =  (1-s4) ;
}

return prob;

}




//****function for generating trans prob matrix when you dont challenge*********


float  transprob_nc(int a, int b)
{
        int d =0, e =0,f=0 ;
        float prob = 0.0 ;

        if((tens(trunclast3(a))== tens(trunclast3(b))) &&
        ((ones(trunclast3(b))- ones(trunclast3(a))) ==1))
                d = 1;
        if((tens(trunclast3(a))== tens(trunclast3(b))) &&
        ((ones(trunclast3(a))- ones(trunclast3(b))) ==1)  &&
        (tens(trunclast3(a))==3) && (ones(trunclast3(a))==4))
                e = 1;
        if((ones(trunclast3(a))== ones(trunclast3(b))) &&
        ((tens(trunclast3(a))- tens(trunclast3(b))) ==1)  &&
         (ones(trunclast3(a))==3) && (tens(trunclast3(a))==4))
                d = 1;
        if((ones(trunclast3(a))== ones(trunclast3(b))) &&
        ((tens(trunclast3(b))- tens(trunclast3(a))) ==1))
                e = 1;
        if ((tens(trunclast3(b))==4) && (tens(trunclast3(a))>= 3) &&
        (ones(trunclast3(a))!=4)&& (ones(trunclast3(b))!=3) &&
        (ones(trunclast3(a))!=tens(trunclast3(a))) )
                e = 1;
        if ((ones(trunclast3(b))==4) && (ones(trunclast3(a))>= 3) &&
         (tens(trunclast3(a))!=4) && (tens(trunclast3(b))!=3)&&
         (ones(trunclast3(a))!=tens(trunclast3(a))) )
                d = 1;
        if ( (ones(trunclast3(a))== 0) && (tens(trunclast3(a))== 4) &&
         (tens(trunclast3(a))== tens(trunclast3(b))) &&
          ((ones(trunclast3(b))== ones( trunclast3(a) )  ) )   )
                f = 1;
```

```
if ( (ones(trunclast3(a))== 4) && (tens(trunclast3(a))== 0) &&
(tens(trunclast3(a))== tens(trunclast3(b))) &&
((ones(trunclast3(b))== ones( trunclast3(a) ) ) )    )
        f = 1;


//when you dont challenge and win

if( e==1 && (hundreds(trunclast3(a)) == 1)  &&
(hundreds(trunclast3(b)) == 1) && (hundreds(b)== hundreds(a)))
        prob = p;


if( e==1 && (hundreds(trunclast3(a)) == 1)  &&
(hundreds(trunclast3(b)) == 2) &&
(hundreds(b)== hundreds(a))&&(hundreds(b)>=1))
{
        if(ones(b) ==1)
        {
                if(tens(b) == 1)
                        prob = p1*(1-p)*k ;
                if(tens(b) == 2)
                        prob = p2*(1-p)*k ;
                if(tens(b) == 3)
                        prob = p3*(1-p)*k ;
                if(tens(b) == 4)
                        prob = p4*(1-p)*k ;
        }

        if(ones(b) ==2)
        {
                if(tens(b) == 1)
                        prob = p1*(1-p)*(1-k) ;
                if(tens(b) == 2)
                        prob = p2*(1-p)*(1-k) ;
                if(tens(b) == 3)
                        prob = p3*(1-p)*(1-k) ;
                if(tens(b) == 4)
                        prob = p4*(1-p)*(1-k) ;
        }

}

if( e==1 && (hundreds(trunclast3(a)) == 1)  &&
(hundreds(trunclast3(b)) == 2) && (hundreds(b)== hundreds(a))&&
(hundreds(b)<1))
{
         prob = (1-p);
}

if( e==1 && (hundreds(trunclast3(a)) == 1)  &&
(hundreds(trunclast3(b)) == 3) && (hundreds(b)== hundreds(a)))
        prob = 1;







//when you dont challenge and lose


if( (d==1) && (hundreds(trunclast3(a)) == 2)  &&
 (hundreds(trunclast3(b)) == 1) && (hundreds(b)== hundreds(a) ) )
        prob = p;

if( (d==1) && (hundreds(trunclast3(a)) == 2)  &&
(hundreds(trunclast3(b)) == 2) && (hundreds(b)== hundreds(a))  &&
(hundreds(b)== hundreds(a))&&(hundreds(b)>=1))
{
        if(ones(b) ==1)
        {
                if(tens(b) == 1)
                        prob = p1*(1-p)*k ;
```

```
                        if(tens(b) == 2)
                                prob = p2*(1-p)*k ;
                        if(tens(b) == 3)
                                prob = p3*(1-p)*k ;
                        if(tens(b) == 4)
                                prob = p4*(1-p)*k ;
                }

                if(ones(b) ==2)
                {
                        if(tens(b) == 1)
                                prob = p1*(1-p)*(1-k) ;
                        if(tens(b) == 2)
                                prob = p2*(1-p)*(1-k) ;
                        if(tens(b) == 3)
                                prob = p3*(1-p)*(1-k) ;
                        if(tens(b) == 4)
                                prob = p4*(1-p)*(1-k) ;
                }

        }

        if( d==1 && (hundreds(trunclast3(a)) == 2)  &&
        (hundreds(trunclast3(b)) == 2) &&
        (hundreds(b)== hundreds(a))&&(hundreds(b)<1))
        {
                 prob = (1-p);
        }

        if( (d==1) && (hundreds(trunclast3(a)) == 2)  &&
         (hundreds(trunclast3(b)) == 3) &&
         (hundreds(b)== hundreds(a)))
                prob = 1;




        //when you loose the challenge



        if( (f==1) && (hundreds(trunclast3(b)) == 3)&&
        (hundreds(trunclast3(a)) == 3) &&  (hundreds(b)== hundreds(a) ) )
        {

                        prob = 1 ;
        }


        return prob;

}







//**********************************main function*****************


int main()
{

        int x = 12345;
        int a,b;


        int y = ones(trunclast3(x));
        int  score[38] = {100, 101, 102, 103, 304,
                                                                110, 111, 112, 113,
                                                                        120, 121, 122, 123,
```

```
                                                                130, 131, 132, 133, 134,
                                                                        340,
                                                200, 201, 202, 203,
                                                        210, 211, 212, 213,
                                                                220, 221, 222, 223,
                                                                        230, 231, 232, 233, 234,


int challengesleft[2]  = {0,1};
int  perception[5]     = {1,2,3,4,5};
int typeofchallenge[3] = {1,2,3};
ofstream CsvFile;

vector<int> states;

        for(int i=0; i <38;i++)
        {
                if(hundreds(score[i])==3)
                {
                        for(int j=0; j<2;j++)
                        {
                                for(int k=0; k<1;k++)
                                {
                                        for(int l=0;l<1;l++)
                                        {
                                                states.push_back(score[i]*1000 +
                                                challengesleft[j]*100 + perception[4]*10
                                                +typeofchallenge[2]) ;
                                        }
                                }
                        }
                }


                if(hundreds(score[i])==1)
                {
                        for(int j=0; j<2;j++)
                        {
                                for(int k=0; k<1;k++)
                                {
                                        for(int l=0;l<1;l++)
                                        {
                                                states.push_back(score[i]*1000 +
                                                challengesleft[j]*100 + perception[4]*10
                                                 +typeofchallenge[2]) ;
                                        }
                                }
                        }
                }

                if(hundreds(score[i])==2)
                {
                        for(int j=0; j<1;j++)
                        {
                                for(int k=0; k<1;k++)
                                {
                                        for(int l=0;l<1;l++)
                                        {
                                                states.push_back(score[i]*1000 +
                                                challengesleft[j]*100 + perception[4]*10
                                                +typeofchallenge[2]) ;
                                        }
                                }
                        }


                        for(int j=0; j<1;j++)
                        {
                                for(int k=0; k<4;k++)
                                {
                                        for(int l=0;l<2;l++)
                                        {
                                                states.push_back(score[i]*1000 +
                                                 challengesleft[j+1]*100 +
                                                 perception[k]*10 +typeofchallenge[l]) ;
```

```
                                        }
                                }
                        }
                }



}

cout<<" the total no of states is :"<<states.size()<<endl;

vector<int> statesc;
for(int i = 0; i<states.size();i++)
{
          if((hundreds(trunclast3(states[i]))==2)  &&
           (hundreds(states[i])>0))
                  statesc.push_back(states[i]);
}

cout<<" the total no of challengable states is :"<<statesc.size()<<endl;

CsvFile.open("allstates.csv");
for(int i = 0; i<states.size();i++)
{
           CsvFile<<states[i]<<endl;
}
CsvFile.close();


CsvFile.open("challengablestates.csv");
for(int i = 0; i<statesc.size();i++)
{
           CsvFile<<statesc[i]<<endl;
}
CsvFile.close();

CsvFile.open("challenge.csv");

       CsvFile<<","<<"c"<<","<<"nc"<<endl;

for(int i = 0; i<states.size();i++)
{
           CsvFile<<states[i]<<",";
           if((hundreds(trunclast3(states[i]))==2)  &&
            (hundreds(states[i])>0))
                  CsvFile<<1<<",";
           else
                   CsvFile<<0<<",";

           CsvFile<<1<<endl;

}
CsvFile.close();


CsvFile.open("reward.csv");

       CsvFile<<","<<"c"<<","<<"nc"<<endl;

for(int i = 0; i<states.size();i++)
{

           CsvFile<<states[i]<<",";
           CsvFile<<0<<",";
           if((hundreds(trunclast3(states[i]))==3)  &&
            (tens(trunclast3(states[i]))==4))
                  CsvFile<<1<<endl;
           else
                   CsvFile<<0<<endl;

}
CsvFile.close();
```

```
CsvFile.open("transprob_c.csv");


CsvFile<<",";

for(int i = 0; i<states.size();i++)
{
        CsvFile<<states[i]<<",";
}
CsvFile<<endl;
for(int i = 0; i<states.size();i++)
{
        CsvFile<<states[i]<<",";
        for(int j = 0; j<states.size();j++)
        {
                CsvFile<<transprob_c(states[i],states[j])<<",";
        }

                CsvFile<<endl;
}

        CsvFile<<endl;
        CsvFile<<endl;
        CsvFile<<endl;
        CsvFile<<endl;
        CsvFile<<endl;
        CsvFile<<endl;
        CsvFile.close();


        CsvFile.open("transprob_nc.csv");

        CsvFile<<",";



for(int i = 0; i<states.size();i++)
{
        CsvFile<<states[i]<<",";
}
CsvFile<<endl;
for(int i = 0; i<states.size();i++)
{
        CsvFile<<states[i]<<",";
        for(int j = 0; j<states.size();j++)
        {
                CsvFile<<transprob_nc(states[i],states[j])<<",";
        }

        CsvFile<<endl;

}
CsvFile.close();







CsvFile.open("transprob.csv");

        CsvFile<<",";



for(int i = 0; i<states.size();i++)
{
        CsvFile<<states[i]<<".c"<<",";
}
for(int i = 0; i<states.size();i++)
{
```

```
                CsvFile<<states[i]<<".nc"<<",";
        }
        CsvFile<<endl;

        for(int i = 0; i<states.size();i++)
        {
                CsvFile<<states[i]<<",";
                for(int j = 0; j<states.size();j++)
                {
                        CsvFile<<transprob_c(states[i],states[j])<<",";
                }
                for(int j = 0; j<states.size();j++)
                {
                        CsvFile<<transprob_nc(states[i],states[j])<<",";
                }

                CsvFile<<endl;

        }
        CsvFile.close();


    return 0;
}
```

Appendix B

Gams code

```
$TITLE OneChallenge
$INLINECOM { }

$OFFUPPER OFFSYMXREF OFFSYMLIST
$onmixed
OPTIONS ITERLIM  = 100000, RESLIM   = 10000, LIMROW   = 0,
        LIMCOL   = 0,      SYSOUT   = OFF,   SOLPRINT = OFF,
        LP = COINCbc, NLP=MINOS5, MIP=cplex ,OPTCR = 0.0001;

SETS
         a actions /c, nc/

         S all states
/
100053
100153
101053
101153
102053
102153
103053
103153
304053
304153
110053
110153
111053
111153
112053
112153
113053
113153
120053
120153
121053
121153
122053
122153
123053
123153
130053
130153
131053
131153
132053
132153
133053
133153
134053
134153
340053
340153
143053
143153
200053
200111
200112
200121
200122
```

200131
200132
200141
200142
201053
201111
201112
201121
201122
201131
201132
201141
201142
202053
202111
202112
202121
202122
202131
202132
202141
202142
203053
203111
203112
203121
203122
203131
203132
203141
203142
210053
210111
210112
210121
210122
210131
210132
210141
210142
211053
211111
211112
211121
211122
211131
211132
211141
211142
212053
212111
212112
212121
212122
212131
212132
212141
212142
213053
213111
213112
213121
213122
213131
213132
213141
213142
220053
220111
220112
220121
220122
220131
220132
220141
220142

221053
221111
221112
221121
221122
221131
221132
221141
221142
222053
222111
222112
222121
222122
222131
222132
222141
222142
223053
223111
223112
223121
223122
223131
223132
223141
223142
230053
230111
230112
230121
230122
230131
230132
230141
230142
231053
231111
231112
231121
231122
231131
231132
231141
231142
232053
232111
232112
232121
232122
232131
232132
232141
232142
233053
233111
233112
233121
233122
233131
233132
233141
233142
234053
234111
234112
234121
234122
234131
234132
234141
234142
243053
243111
243112
243121

```
243122
243131
243132
243141
243142 /
          ;


Alias (S,S1);


Table

C(S,S1)  transition probability matrix when you challenge

$ondelim
$include  transprob_c.csv
$offdelim


Table

NC(S,S1)  transition probability matrix when you challenge

$ondelim
$include  transprob_nc.csv
$offdelim



$ontext

Table

P(S1,S,a)  transition probability matrix when you challenge
$ondelim
$include  transprob.csv
$offdelim

$offtext

Table

I(S,a) indicates whether an action 'a' is available in state 'S'

$ondelim
$include  challenge.csv
$offdelim

Table

g(S,a) reward obtained when you take an action a in state S

$ondelim
$include  reward.csv
$offdelim

;


Display C;
Display NC;




POSITIVE VARIABLES
  q(S,a)     proportion of time you take action a in recurrent state S
  r(S,a)     prop of time you take action a in transient state S ;

FREE VARIABLE
  TOTALREW     total probability of winning ;
```

```
EQUATIONS
   OBJ                 defines the objective function - maximize total reward earned
   BELLMAN1(S)         corresponds to first equation of the set of bellman equations
   BELLMAN2(S)         corresponds to second equation of the set of bellman equations ;

OBJ.. TOTALREW =E= SUM((S,a)$(I(S,a)=1),q(S,a)*g(S,a));

BELLMAN1(S)..    SUM(a$(I(S,a)=1),q(S,a)) =E= SUM(S1$(I(S1,"c")=1) , q(S1,"c")*C(S1,S) ) +
 SUM(S1$(I(S1,"nc")=1) , q(S1,"nc")*NC(S1,S) ) ;
BELLMAN2(S)..    SUM(a$(I(S,a)=1),q(S,a) + r(S,a)) =E= (1/card(S)) +
SUM(S1$(I(S1,"c")=1) , r(S1,"c")*C(S1,S) ) + SUM(S1$(I(S1,"nc")=1) , r(S1,"nc")*NC(S1,S) ) ;



MODEL Onechallenge /ALL/ ;

SOLVE Onechallenge USING LP MAXIMIZING TOTALREW;

$ontext
DISPLAY TOTALREW.L ,q.L,r.L ;
$offtext

file results /results.txt/;
put results;
put "Model status",OneChallenge.modelstat/;
put "Solver status",OneChallenge.solvestat/;

put "Totalreward", TOTALREW.L/;
put "recurrent"/;
loop((S,a) $(q.l(S,a) > 0.0) ,
   put S.tl,a.tl,q.l(S,a)/
);

put "transient"/;
loop((S,a) $(r.l(S,a) > 0.0),
   put S.tl,a.tl,r.l(S,a)/
);

putclose;
```

Bibliography

[1] P. K. Newton and J. B. Keller, "Probability of winning at tennis I. theory and data," *Studies in Applied Mathematics*, vol. 114, no. 3, pp. 241–269, 2005.

[2] P. K. Newton and A. Kamran, "Monte carlo tennis: A stochastic Markov chain model," *Journal of Quantitative Analysis in Sports*, vol. 5, no. 3, 2009.

[3] A. J. O'Malley, "Probability formulas and statistical anlaysis in tennis," *Journal of Quantitative Analysis in Sports*, vol. 4, no. 2, 2008.

[4] J. M. Norman, "Dynamic programming in tennis - when to use a fast serve," *Journal of the Operational Research Society*, vol. 36, no. 1, pp. 75–77, 1985.

[5] S. R. Clarke, "Dynamic programming in one-day cricket - optimal scoring rates," *Journal of the Operational Research Society*, vol. 39, no. 4, pp. 331–337, 1988.

[6] D. Kohler, "Optimal strategies for the game of darts," *Journal of the Operational Research Society*, vol. 33, pp. 871–884, 1982.

[7] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, 3rd ed. Athena Scientific, Belmont, Massachusetts, 2007, vol. 2.

[8] ——, *Dynamic Programming and Optimal Control*, 3rd ed. Athena Scientific, Belmont, Massachusetts, 2005, vol. 1.

[9] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, 1st ed. Wiley-Interscience, 2005.

Vita

Vamsi Krishna Nadimpalli was born in Andhra Pradesh, India on April 28, 1986, the son of Rama Krishnam Raju Nadimpalli and Krishna Veni Nadimpalli. He received the Bachelor of Technology degree from the Indian Institute of Technology, Roorkee in 2008. He began his graduate studies in the department of Operations Research and Industrial Engineering at the University of Texas at Austin in the fall of 2008. He joined the group of Dr. John Hasenbein in fall 2009 to begin his Masters thesis work.

Permanent address: vamsi.k.nadimpalli@gmail.com

This thesis was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.